JOURNAL OF COMPUTER AND SYSTEM SCIENCES 16, 67–99 (1978)

IO and OI. II

JOOST ENGELFRIET* AND ERIK MEINECHE SCHMIDT

Aarhus University, Aarhus, Denmark

Received August 6, 1975; revised May 1, 1977

In Part 1 of this paper (*J. Comput. System Sci.* **15**, Number 3 (1977)) we presented a fixed point characterization of the (IO and OI) context-free tree languages. We showed that a context-free tree grammar can be viewed as a system of regular equations over a tree language substitution algebra. In this part we shall use these results to obtain a theory of systems of context-free equations over arbitrary continuous algebras. We refer to the Introduction of Part 1 for a description of the contents of this part.

5. SYSTEMS OF CONTEXT-FREE EQUATIONS OVER ARBITRARY Σ-ALGEBRAS (OR: NONDETERMINISTIC RECURSIVE PROGRAM SCHEMES)

In this section we shall view a context-free tree grammar as a computational device to define subsets and relations over an arbitrary (possibly nondeterministic) Σ-algebra, in other words, as a nondeterministic recursive program scheme with the Σ-algebra playing the role of an interpretation of the program scheme. One can think of these program schemes as similar to the usual ones (see, for instance, [18, 20, 25, 28]), but without tests and with expressions like τ_1 *or* τ_2 to be evaluated as τ_1 or as τ_2 nondeterministically. Thus, nondeterministic recursive program schemes compute relations rather than functions. As an example, let $\Sigma_0 = \{a\}$, $\Sigma_1 = \{f\}$ and let G be the context-free tree grammar $(\Sigma, \{S, F\}, P)$ where P consists of the productions $F(x) \to x$, $F(x) \to F(f(x))$, and $S \to F(a)$. Then F can be considered as the recursive program scheme $F(x) = x$ *or* $F(f(x))$, and S as a call of F with some input a. In the Σ-algebra with domain $\mathbb{N}$ and $f_{\mathbb{N}}(x) = x + 1$, F computes the relation $\{(x, y) \mid y \geqslant x\}$ and S computes a set depending on the input $a_{\mathbb{N}}$.

However, rather than defining the computation of a context-free tree grammar in a Σ-algebra (using a specific computation rule) and characterizing the computed relation as the solution of a system of equations (as we did in Section 3), we take the shortcut of just considering the context-free grammar as a system of "context-free" equations, leaving it to the reader to be convinced of the computational naturalness of this definition (thus we rely on fixed point semantics for our nondeterministic recursive program schemes).

The aim of this section is to find "Mezei- and Wright-like" (abbreviated by *MW-like*) results for context-free tree grammars, i.e., we want to find a "tree algebra," preferably

* Present address: Twente University of Technology, Enschede, Netherlands.

0022-0000/78/0161-0067$02.00/0

some $\mathscr{P}(T_\Sigma(X))$, such that the solution of a system of context-free equations over any Σ-algebra is the homomorphic image of its solution in that tree algebra. Intuitively such an MW-like result means that, instead of computing in the Σ-algebra, one may as well do the computation "symbolically," i.e., on trees, and afterwards interpret the result in the Σ-algebra. (The "Herbrand theorem" in program scheme theory is a result in this direction.)

There are two well-known modes of computation for (nondeterministic) recursive programs: call by value and call by name (see [19]). In a *call by value* computation the actual parameters of a function call have to be values from the domain of computation. Hence, to obtain the relation computed in the call by value mode by a context-free tree grammar in a Σ-algebra, it is natural to consider the grammar as a system of equations to be solved in the algebra of relations over the Σ-algebra, and to use composition of relations as the basic operation in these equations (see [25]). In a *call by name* computation the actual parameters of a function call are formal expressions which stand for (possibly empty) sets of values from the domain of computation (each element of such a set being a possible value of the actual parameter). Hence, to model the call by name computation of a context-free tree grammar in a Σ-algebra, it seems natural to solve the grammar in the algebra of functions of subsets of the Σ-algebra, and to use composition of these functions as the basic operation (cf. [8]). The relation computed by the grammar is then obtained by restricting the subset function to singletons. Note that for deterministic program schemes the set of possible values of an actual parameter is always a singleton or empty. In this case it suffices to add an element ω (standing for "undefined," or the empty set) to the domain and to consider functions over the so extended domain (see [18, 28]).

To define the solution of a context-free tree grammar G as a system of context-free equations in an algebra of relations or functions (over some Σ-algebra) we could proceed along the same lines as in Section 3 for tree languages, using composition of relations or functions rather than substitution of tree languages. Then, clearly, the solution of G would equal the solution of the corresponding system G^D of regular $D(\Sigma)$-equations (cf. Section 4). Therefore we shall just define the proper (Δ-continuous) $D(\Sigma)$-algebras of relations or functions and consider solutions of regular $D(\Sigma)$-equations in these algebras.

5.1. DEFINITION. A context-free tree grammar G with terminal alphabet Σ will be called a *system of context-free Σ-equations*. The *solution* of G in a Δ-continuous $D(\Sigma)$-algebra A (with $\sqcup$-complete carriers) is defined to be the solution of the system G^D of regular $D(\Sigma)$-equations in A (see Section 4). ∎

Now Lemma 4.8 shows that we may even, without increase of generality, consider arbitrary systems of regular $D(\Sigma)$-equations rather than just those obtained from context-free tree grammars (provided (*) of the lemma holds).

Thus, for a given Σ-algebra A and a system of regular $D(\Sigma)$-equations, depending on whether we solve the system in the algebra of relations over A or the algebra of functions over subsets of A, we obtain a call by value solution and a call by name solution over A, respectively. Our MW-like results will relate the call by value solution to an IO tree

language (in fact, the solution in $\mathscr{P}(T_\Sigma(X))_{IO}$) or a recognizable tree language (the solution in $\mathscr{P}(T_{D(\Sigma)})$), and the call by name solution to an OI tree language (in fact, the solution in $\mathscr{P}(T_\Sigma(X))_{OI}$). To obtain these results it suffices to show the existence of a ($\sqcup$-continuous) $D(\Sigma)$-homomorphism from the tree language algebra into the algebra of relations or functions, as shown by the following lemma of which the easy proof is left to the reader (cf. Lemma 5.3 of [21]).

5.2. LEMMA. *Let Σ be an S-sorted alphabet. Let A and B be Δ-continuous Σ-algebras with $\sqcup$-complete carriers. Let h* be a *$\sqcup$-continuous Σ-homomorphism $A \to B$. Then h preserves solutions of regular Σ-equations* (i.e., *if E is a system of regular Σ-equations, then $h(|\, EA\,|_i) = |\, EB\,|_i$ for all i*). ▮

We note here that we will in general be interested in nonterministic Σ-algebras, i.e., domains of computation in which the basic operations are possibly partial or many valued.

The rest of this section is organized as follows. First we discuss the IO case. Then the OI case is treated. Finally we look at some connections with the literature, in particular deterministic program schemes with tests (cf. [18]) and the MW-like results for infinite trees in [15] and [23].

Let us start by considering *the inside-out or call by value case*. Let Σ be a fixed ranked alphabet and D its derived alphabet.

We first define the algebra of relations over a (nondeterministic) Σ-algebra.

5.3. DEFINITION. Let A be a nondeterministic Σ-algebra. We define $\mathscr{R}(A)$, the *D-algebra of relations over A*, as follows:

(i) for $n \geqslant 0$, $\mathscr{R}(A)_n = \mathscr{P}(A^{n+1})$;

(ii) for $n \geqslant 0$ and $f \in \Sigma_n$, $f' = f_A$ (more precisely

$$f' = \{(a_1, \ldots, a_n, a) \mid a \in f_A(a_1, \ldots, a_n)\});$$

(iii) for $n \geqslant 1$ and $1 \leqslant i \leqslant n$,

$$\pi_i^n = \{(a_1, \ldots, a_n, a_i) \mid a_1, \ldots, a_n \in A\},$$

i.e., π_i^n is the ith projection $A^n \to A$;

(iv) for n, $k \geqslant 0$, $R \subseteq A^{n+1}$ and $R_1, \ldots, R_n \subseteq A^{k+1}$, $c_{n,k}(R, R_1, \ldots, R_n) = R \circ (R_1, \ldots, R_n)$, i.e., $c_{n,k}$ is composition of relations (in fact, $R \circ (R_1, \ldots, R_n) = \{(a_1, \ldots, a_k, a) \mid$ there are $b_1, \ldots, b_n \in A$ such that $(a_1, \ldots, a_k, b_i) \in R_i$ and $(b_1, \ldots, b_n, a) \in R\}$; $c_{0,k}(R) = \{(a_1, \ldots, a_k, a) \mid a \in R\}$). ▮

Note that $\mathscr{R}(A)_0 = \mathscr{P}(A)$.

The easy proof of the following lemma is left to the reader.

5.4. LEMMA. *For every nondeterministic Σ-algebra A, $\mathscr{R}(A)$ is a $\sqcup$-continuous D-algebra (with set inclusion as ordering).* ▮

Observe that $\mathscr{R}(A)$ is in fact the subset algebra of the nondeterministic D-algebra with A^{n+1} as domain of sort n and the obvious (partial) $c_{n,k}$ operations on tuples.

In accordance with the discussion at the beginning of this section we now define (context-free) equational relations over a Σ-algebra.

5.5. DEFINITION. Let A be a nondeterministic Σ-algebra. For $n \geqslant 0$, a relation $R \subseteq A^{n+1}$ is said to be IO *equational* if it is equational as an element of the $\sqcup$-continuous D-algebra $\mathscr{R}(A)$ (i.e., iff there is a system E of regular D-equations such that $R = |E\mathscr{R}(A)|_i$ for some i). ▮

Note that in particular ($n = 0$) subsets of A may be IO equational.

The following MW-like result is now immediate.

5.6. THEOREM. *Let A be a nondeterministic Σ-algebra and E a system of regular D-equations. Then any component of the solution of E in $\mathscr{R}(A)$ is the D-homomorphic image of the corresponding component of the solution of E in $\mathscr{P}(T_D)$.*

Proof. Since there is a (unique) $\sqcup$-continuous D-homomorphism from $\mathscr{P}(T_D)$ into $\mathscr{R}(A)$ (see Theorem 2.3.5), the theorem follows from Lemma 5.2. ▮

5.7. COROLLARY. *Let A be a nondeterministic Σ-algebra. An element of $\mathscr{R}(A)$ is* IO *equational iff it is the D-homomorphic image of a recognizable tree language in T_D.*

Proof. Immediate from the previous theorem and the fact that in $\mathscr{P}(T_D)$ the equational elements are the recognizable tree languages (see Section 4). ▮

Given a system of context-free Σ-equations $G = (\Sigma, \mathscr{F}, P)$ and $F \in \mathscr{F}_n$, the component corresponding to F in the solution of G in $\mathscr{R}(A)_n$ might be called the *call by value relation computed by (G, F) over A*. Thus, by Theorem 5.6, the call by value relation computed by (G, F) is the D-homomorphic image of the recognizable tree language generated by nonterminal F in T_D (viewing G^D as a regular tree grammar in the obvious way). From the computational point of view this means that, instead of computing in A with some input $(a_1, \ldots, a_n)$, one can, nondeterministically, generate a tree in T_D, interpret it as a relation, and find an element $(a_1, \ldots, a_n, a)$ in this relation. Then the element a is one of the possible outputs.

We now ask whether an IO equational relation can be obtained from the context-free grammar by first computing formally with Σ-trees, i.e., generating an IO tree language with variables (where intuitively the variables stand for the input values), and then interpreting the tree language with variables as a relation. The obvious way to interpret such a tree language as a relation is defined as follows

5.8. DEFINITION. Let A be a nondeterministic Σ-algebra. Let $n \geqslant 0$ and $L \subseteq T_\Sigma(X_n)$. The *derived relation* of L over A, denoted by L^A or $\mathrm{derrel}_A(L)$, is the relation in A^{n+1} defined by $L^A = \{(a_1, \ldots, a_n, b) \mid b \in \bar{a}(L)$, where $\bar{a}$ is the unique $\sqcup$-continuous Σ-homomorphism $\mathscr{P}(T_\Sigma(X_n)) \to \mathscr{P}(A)$ such that $\bar{a}(x_i) = \{a_i\}\}$. ▮

It is easy to see that in the case that A is an ordinary (deterministic) Σ-algebra, $L^A = \{(a_1, \ldots, a_n, a) \mid t_A(a_1, \ldots, a_n) = a \text{ for some } t \text{ in } L\}$.

We now show that in the case of an ordinary Σ-algebra an MW-like result, as indicated above, can be obtained.

5.9. THEOREM. *Let A be a Σ-algebra. The mapping* derrel_A *is the unique* $\sqcup$*-continuous D-homomorphism* $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}} \to \mathscr{R}(A)$.

Proof.

$$\begin{array}{ccc} & T_D & \\ & \downarrow \text{YIELD} & \\ DT_\Sigma(X) & \subseteq & \mathscr{P}(T_\Sigma(X))_{\mathrm{IO}} \\ \downarrow \mathrm{derop}_A & & \downarrow \mathrm{derrel}_A \\ \mathscr{F}(A) & \subseteq & \mathscr{R}(A) \end{array}$$

Consider first the mapping $\mathrm{derop}_A: DT_\Sigma(X) \to \mathscr{R}(A)$, where we identify a function $A^n \to A$ with its graph in $\mathscr{P}(A^{n+1})$. This mapping is a D-homomorphism (see the end of Section 2.2). Moreover, since the (unique) D-homomorphism YIELD: $T_D \to DT_\Sigma(X)$ is onto (in fact, for $t \in T_\Sigma(X)$, $t = \mathrm{YIELD}(\mathrm{COMB}(t))$, where COMB: $T_\Sigma(X) \to T_D$ is defined in Definition 4.4), derop_A is the unique D-homomorphism $DT_\Sigma(X) \to \mathscr{R}(A)$. Hence by (the proof of) Lemma 2.3.4 and the fact that, for $L \subseteq T_\Sigma(X)$, $L^A = \bigcup_{t \in L} t^A$, derrel_A is the unique $\sqcup$-continuous D-homomorphism extending derop_A (recall that $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}}$ is the subset algebra of $DT_\Sigma(X)$). Since the restriction to $DT_\Sigma(X)$ of any D-homomorphism $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}} \to \mathscr{R}(A)$ is a D-homomorphism, derrel_A is unique. ∎

From this theorem and Lemma 5.2 we directly obtain the following MW-like result for IO context-free tree grammars.

5.10. THEOREM. *Let A be a Σ-algebra and E a system of regular D-equations. Then any component of the solution of E in $\mathscr{R}(A)$ is the derived relation of the corresponding component of the solution of E in $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}}$ (in formula:* $|E\mathscr{R}(A)|_i = \mathrm{derrel}_A(|E\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}}|_i)$ *for all i).* ∎

Using Corollary 4.10 we have the following corollary.

5.11. COROLLARY. *Let A be a Σ-algebra. An element of $\mathscr{R}(A)$ is* IO *equational if and only if it is the derived relation of an* IO *tree language with variables.* ∎

Note in particular that a subset of A is IO equational iff it is the Σ-homomorphic image of an IO tree language over Σ (thus a subset of T_Σ is IO equational iff it is an IO Σ-tree language).

Corollary 5.11 may be stated more precisely as follows. Let $G = (\Sigma, \mathscr{F}, P)$ be a system of context-free Σ-equations and $F \in \mathscr{F}_n$. Then the call by value relation computed by (G, F) over A is the derived relation of the IO tree language with variables $L_{\mathrm{IO}}(G, F(x_1 \cdots x_n))$.

Clearly, when solving D-equations in $\mathscr{R}(A)$, we may restrict attention to derived relations.

5.12. DEFINITION. Let A be a Σ-algebra. We define the *D-algebra of derived relations over* A, denoted by $\mathrm{der}\mathscr{R}(A)$, to be the image of $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}}$ in $\mathscr{R}(A)$ under the mapping derrel_A. ∎

Thus, for $n \geqslant 0$, $\mathrm{der}\mathscr{R}(A)_n = \{L^A \mid L \subseteq T_\Sigma(X_n)\}$.

Obviously $\mathrm{der}\mathscr{R}(A)$ is a $\sqcup$-continuous sub$D$-algebra of $\mathscr{R}(A)$ (in fact the smallest one), and therefore the solution of any system of regular D-equations in $\mathscr{R}(A)$ is equal to its solution in $\mathrm{der}\mathscr{R}(A)$.

As a special case, let us consider the Σ-algebra T_Σ. Clearly the mapping $\mathrm{derrel}_{T_\Sigma}$: $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}} \to \mathrm{der}\mathscr{R}(T_\Sigma)$ is the identity for sort 0 (i.e., for elements of $\mathscr{P}(T_\Sigma)$). Hence, given an IO tree grammar $G = (\Sigma, \mathscr{F}, P)$ and an $S \in \mathscr{F}_0$, the component corresponding to S is the same in the solutions in $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}}$ and $\mathrm{der}\mathscr{R}(T_\Sigma)$. Thus we have obtained a fixed point characterization of IO tree languages in the space $\mathrm{der}\mathscr{R}(T_\Sigma)$. For $F \in \mathscr{F}_n$ $(n \geqslant 1)$ it follows from previous remarks that its solution in $\mathrm{der}\mathscr{R}(T_\Sigma)$ is $\mathrm{derrel}_{T_\Sigma}(L_{\mathrm{IO}}(G, F(x_1 \cdots x_n)))$ which is (using an obvious property of $\overset{*}{\underset{\mathrm{IO}}{\Rightarrow}}$) equal to $\{(t_1, \ldots, t_n, t) \mid F(t_1 \cdots t_n) \overset{*}{\underset{\mathrm{IO}}{\Rightarrow}} t\}$.

Note that, in contrast to the case of trees (see the end of Section 2.2), the algebras $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}}$ and $\mathrm{der}\mathscr{R}(T_\Sigma)$ are *not* isomorphic. As an example, in $\mathscr{P}(T_\Sigma(X_1))$, $\mathrm{derrel}_{T_\Sigma}(T_\Sigma) = \mathrm{derrel}_{T_\Sigma}(T_\Sigma \cup \{x_1\}) = T_\Sigma \times T_\Sigma$.

Consider now the free Σ-algebra $T_\Sigma(X)$ with generators X. Remarks analogous to those for T_Σ also hold for $T_\Sigma(X)$. Moreover the algebras $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}}$ and $\mathrm{der}\mathscr{R}(T_\Sigma(X))$ *are* isomorphic. Thus, by Theorem 5.9, $T_\Sigma(X)$ is a "universal interpretation": two IO tree grammars G_1 and G_2, with nonterminals F_1 and F_2, compute the same call by value relation over all Σ-algebras iff they do so over $T_\Sigma(X)$ (this was brought to the attention of the authors by M. Nivat).

For a nondeterministic Σ-algebra A both Theorems 5.9 and 5.10 break down in general: there is no homomorphism $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}} \to \mathscr{R}(A)$. As a first example, let $f \in \Sigma_2$ and $b \in \Sigma_0$, and suppose that $b_A = \{a_1, a_2\}$ and f_A is a total function $A^2 \to A$. Consider the IO tree grammar G with productions $S \to F(b)$ and $F(x_1) \to f(x_1x_1)$. Then the Σ-homomorphic image of the language $\{f(bb)\}$ generated by G from S is in general not equal to the solution of S in $\mathscr{R}(A)$. In fact, $\{f(bb)\}^A = \{f_A(a_1, a_1), f_A(a_1, a_2), f_A(a_2, a_1), f_A(a_2, a_2)\}$, but the solution of S in $\mathscr{R}(A)$ is $\{f_A(a_1, a_1), f_A(a_2, a_2)\}$. The reason for this failure is obviously that during call by value computation of $F(b)$ in A we have to fix a value a_1 or a_2 of b before copying it. This process cannot be mirrored in the derivation of the tree grammar. Thus we cannot compute symbolically on trees, but we have to "consult the interpretation" during computation. As a second example, let $d \in \Sigma_0$ and $p \in \Sigma_1$ and suppose that $d_A = \{a\}$ and $p_A: A \to A$ is a partial function such that $p_A(a)$ is undefined. Consider the IO tree grammar with productions $S \to F(dp(d))$ and $F(x_1x_2) \to x_1$. Then the language generated is $\{d\}$, but the solution of S in $\mathscr{R}(A)$ is $\varnothing$. Again we cannot just compute with tress because we have to test whether $p_A(d_A)$ has a value before deleting the tree $p(d)$. Thus, in a nondeterministic Σ-algebra, the only way

to do a symbolic computation seems to be computing with trees in T_D (see Theorem 5.6), which keep all information about the copying and deletion (which have to be done after the symbolic computation).

One should observe that the above two examples are essentially the same as the failure of the associativity law and the projection laws in $\mathscr{P}(T_\Sigma(X))_{\mathrm{IO}}$ (see Section 2.4). For the first example,

$$(f(x_1x_2) \underset{\mathrm{IO}}{\leftarrow} (b, b)) \underset{\mathrm{IO}}{\leftarrow} \{a_1, a_2\} \neq f(x_1x_2) \underset{\mathrm{IO}}{\leftarrow} (b \underset{\mathrm{IO}}{\leftarrow} \{a_1, a_2\}, b \underset{\mathrm{IO}}{\leftarrow} \{a_1, a_2\}),$$

where b is used to denote x_1. For the second example, $x_1 \underset{\mathrm{IO}}{\leftarrow} (d, \varnothing) \neq d$.

Let us now turn to *the outside-in or call by name case.* Let Σ again be a fixed ranked alphabet and D its derived alphabet. Instead of considering the subset Σ-algebra $\mathscr{P}(A)$ corresponding to some nondeterministic Σ-algebra A, we shall prove our results for the slightly more general case of a $\bigcup$-continuous Σ-algebra. For later use the first few definitions will be given for a Δ-continuous Σ-algebra with $\bigcup$-complete carrier.

5.13. DEFINITION. Let B be a Δ-continuous Σ-algebra with $\bigcup$-complete carrier. We define $\mathscr{F}_\Delta(B)$, the *D-algebra of Δ-continuous functions over B*, as follows:

(i) for $n \geqslant 0$, $\mathscr{F}_\Delta(B)_n$ is the set of all Δ-continuous total functions $B^n \to B$ (for $n = 0$, $\mathscr{F}_\Delta(B)_0 = B$);

(ii) for $n \geqslant 0$ and $f \in \Sigma_n$, $f' = f_B$;

(iii) for $n \geqslant 1$, $1 \leqslant i \leqslant n$ and $b_1, \ldots, b_n \in B$, $\pi_i^n(b_1, \ldots, b_n) = b_i$, i.e., π_i^n is the ith projection $B^n \to B$;

(iv) for n, $k \geqslant 0$, $f \in \mathscr{F}_\Delta(B)_n$ and $g_1, \ldots, g_n \in \mathscr{F}_\Delta(B)_k$, $c_{n,k}(f, g_1, \ldots, g_n) = f \circ (g_1, \ldots, g_n)$, i.e., $c_{n,k}$ is composition of functions ($c_{0,k}(a)$ is the constant a: $B^k \to B$, i.e., $c_{0,k}(a)(b_1, \ldots, b_k) = a$ for all $b_1, \ldots, b_k \in B$).

Moreover, each carrier $\mathscr{F}_\Delta(B)_n$ is ordered in the usual way: $f \subseteq g$ if and only if $f(b_1, \ldots, b_n) \subseteq g(b_1, \ldots, b_n)$ for all $b_1, \ldots, b_n \in B$. ▮

It is left to the reader to verify the correctness of the definition (the projections are Δ-continuous and composition preserves Δ-continuity). The straightforward proof of the next lemma is also left to the reader.

5.14. LEMMA. *For every Δ-continuous Σ-algebra B with $\bigcup$-complete carrier, $\mathscr{F}_\Delta(B)$ is a Δ-continuous D-algebra with $\bigcup$-complete carriers.* ▮

Note that $\mathscr{F}_\Delta(B)$ is a sub$D$-algebra of $\mathscr{F}(B)$, and hence a "subclone" of $\mathscr{F}(B)$ (cf. the end of Section 2.2). Thus $\mathscr{F}_\Delta(B)$, and each of its Δ-complete sub$D$-algebras, might be called a "Δ-continuous clone." Every Δ-continuous clone is then a μ-clone in the sense of [40], but not vice versa. In fact the smallest μ-clone in $\mathscr{F}_\Delta(B)$, denoted by $\mu\mathrm{Cl}(B)$ in [4], is equal to $\{L_B \mid L \text{ is a recognizable tree language with variables}\}$ (see Definition 2.3.7).

We now define (context-free) equational functions over a Δ-continuous Σ-algebra (cf. the discussion in the beginning of this section).

5.15. DEFINITION. Let B be a Δ-continuous Σ-algebra with $\sqcup$-complete carrier. For $n \geqslant 0$, a (Δ-continuous) function $f: B^n \to B$ is said to be OI *equational* if it is equational as an element of the Δ-continuous D-algebra $\mathscr{F}_\Delta(B)$. ▮

Note that in particular ($n = 0$) elements of B may be OI equational.

We now turn to an MW-like result for $\sqcup$-continuous Σ-algebras, in particular subset algebras of nondeterministic Σ-algebras. It will turn out that an OI equational function can be obtained by interpreting an OI tree language with variables as a function. The obvious way to do this is by taking its derived operation in the $\sqcup$-continuous Σ-algebra (see Definition 2.3.7).

5.16. THEOREM. *Let B be a $\sqcup$-continuous Σ-algebra. The mapping* derop_B *is the unique $\sqcup$-continuous D-homomorphism* $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}} \to \mathscr{F}_\Delta(B)$.

Proof. Obviously $\mathrm{derop}_B(f(x_1 \cdots x_n)) = f_B$ and $\mathrm{derop}_B(x_i)$ is the ith projection. It now follows from Theorem 2.4.1 that derop_B is a D-homomorphism. Moreover derop_B is clearly $\sqcup$-continuous. It is easy to see that there can at most be one $\sqcup$-continuous D-homomorphism from $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}}$ into any continuous algebra. ▮

From this theorem and Lemma 5.2 we immediately obtain the following MW-like result for systems of OI context-free equations.

5.17. THEOREM. *Let B be a $\sqcup$-continuous Σ-algebra and E a system of regular D-equations. Then any component of the solution of E in $\mathscr{F}_\Delta(B)$ is the derived operation of the corresponding component of the solution of E in $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}}$ (in formula:* $|E\mathscr{F}_\Delta(B)|_i = \mathrm{derop}_B(|E\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}}|_i)$ *for all i).* ▮

Using Corollary 4.10 we have the following corollary.

5.18. COROLLARY. *Let B be a $\sqcup$-continuous Σ-algebra. An element of $\mathscr{F}_\Delta(B)$ is* OI *equational iff it is the derived operation of an* OI *tree language with variables.* ▮

Note in particular that an element of B is OI equational iff it is the Σ-homomorphic image of an OI tree language over Σ (thus a subset of T_Σ is OI equational iff it is an OI Σ-tree language).

Note also that, analogous to the IO case, one may restrict attention to $\mathrm{der}\mathscr{F}_\Delta(B)$, the *$D$-algebra of derived operations over B.*

Note finally that in the case of a Δ-continuous Σ-algebra B, both Theorems 5.16 and 5.17 fail to hold in general: there is no homomorphism $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}} \to \mathscr{F}_\Delta(B)$. As an example, let B have domain $\mathscr{P}(A)$ for some set A. Let $a \in \Sigma_0$ and $or \in \Sigma_2$. Let a_B be some nonempty subset of A and let or_B: $B^2 \to B$ be union of sets, i.e., $or_B(A_1, A_2) = A_1 \cup A_2$. Note that or_B is Δ-continuous, but not $\sqcup$-continuous in its arguments (it fails on $\sqcup \varnothing$). Consider the OI tree grammar G with productions $S \to F(Q)$, $F(x_1) \to or(ax_1)$, and $Q \to Q$. Then the solution of S in $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}}$ is $\varnothing$, but its solution in $\mathscr{F}_\Delta(B)$ is a_B.

For a given system of context-free Σ-equations $G = (\Sigma, \mathscr{F}, P)$ with $F \in \mathscr{F}_n$ and a nondeterministic Σ-algebra A, one is often not interested in the solution of F in $\mathscr{F}_\Delta(\mathscr{P}(A))$ as a function $\mathscr{P}(A)^n \to \mathscr{P}(A)$, but in the restriction of this function to singletons.

5.19. DEFINITION. Let $G = (\Sigma, \mathscr{F}, P)$ be an OI tree grammar, $F \in \mathscr{F}_n$ and A a nondeterministic Σ-algebra. The *call by name relation computed by* (G, F) *over* A is defined to be the relation $\{(a_1, \ldots, a_n, a) \mid a \in f(\{a_1\}, \ldots, \{a_n\})\}$, where f is the component corresponding to F in the solution of G in $\mathscr{F}_\Delta(\mathscr{P}(A))$. ∎

It follows easily from the definitions of derived operation and derived relation that, for any nondeterministic Σ-algebra A and any $L \subseteq T_\Sigma(X)$, the restriction of $L_{\mathscr{P}(A)}$ to singletons is L^A. Hence, by Theorem 5.17, the call by name relation computed by (G, F) over A equals the derived relation of $L_{\mathrm{OI}}(G, F(x_1 \cdots x_n))$ over A.

As a special case, let us consider the Σ-algebra T_Σ. Clearly the mapping $\mathrm{derop}_{\mathscr{P}(T_\Sigma)}$: $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}} \to \mathscr{F}_\Delta(\mathscr{P}(T_\Sigma))$ is the identity for elements of $\mathscr{P}(T_\Sigma)$. Hence, given an OI tree grammar $G = (\Sigma, \mathscr{F}, P)$, the component corresponding to S is the same for the solutions in $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}}$ and $\mathscr{F}_\Delta(\mathscr{P}(T_\Sigma))$. Thus we obtain an alternative fixed point characterization of the OI tree languages, which is due to Downey [8]. For $F \in \mathscr{F}_n$ $(n \geqslant 1)$ it follows from previous remarks that the call by name relation computed by (G, F) over T_Σ is $\mathrm{derrel}_{T_\Sigma}(L_{\mathrm{OI}}(G, F(x_1 \cdots x_n)))$ which is (using an obvious property of $\overset{*}{\underset{\mathrm{OI}}{\Rightarrow}}$) equal to $\{(t_1, \ldots, t_n, t) \mid F(t_1 \cdots t_n) \Rightarrow_{\mathrm{OI}}^* t\}$. Note that, by the same example as in the IO case, the algebras $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}}$ and $\mathrm{der}\mathscr{F}_\Delta(\mathscr{P}(T_\Sigma))$ are not isomorphic. Consider now the Σ-algebra $T_\Sigma(X)$. Remarks analogous to those for T_Σ also hold for $T_\Sigma(X)$. The algebras $\mathscr{P}(T_\Sigma(X))_{\mathrm{OI}}$ and $\mathrm{der}\mathscr{F}_\Delta(\mathscr{P}(T_\Sigma(X)))$ are isomorphic. Moreover it is easy to see that $T_\Sigma(X)$ is a "universal interpretation": two OI tree grammars G_1 and G_2, with nonterminals F_1 and F_2, compute the same call by name relation over all (nondeterministic) Σ-algebras iff they do so over $T_\Sigma(X)$.

In the rest of this section we look at some connections with the literature. We shall first show how *ordinary recursive program (scheme)s* (see [18, 5.2]) fit into our formalism in both the call by value and the call by name case (the difference being that ordinary recursive program schemes have tests whereas ours have nondeterminism). The main problem is the representation of the *if–then–else* construction. We shall use the well-known "trick" of representing a conditional expression like *if* $p(x)$ *then* $f(x)$ *else* $g(x)$ by the nondeterministic expression

$$(\textit{if } p(x) \textit{ then } f(x) \textit{ else } \omega) \textit{ or } (\textit{if } p(x) \textit{ then } \omega \textit{ else } g(x)),$$

where ω stands for "undefined." Moreover, to be able to represent the components of this expression, we introduce first a basic function pr_2 which stands for the second projection, and second, to replace $p(x)$, two partial functions $p_Y(x)$ and $p_N(x)$, which are defined iff $p(x)$ is true and false, respectively, and, if defined, deliver some arbitrary value (for instance, x). Our expression is now representable as $pr_2(p_Y(x), f(x))$ *or* $pr_2(p_N(x), g(x))$. The recursive program $F(x, y) =$ *if* $x = 0$ *then* 1 *else* $F(x - 1, F(x, y))$ will, for instance, be represented as follows: let $z(x)$ stand for $x = 0$, $f(x)$ for $x - 1$, and a for 1, then the representation is $F(x, y) = pr_2(z_Y(x), a)$ *or* $pr_2(z_N(x), F(f(x), F(x, y)))$.

We shall now state this more formally. Let a recursive program scheme S consist of a finite ranked alphabet $\mathscr{F}$ of function symbols, a finite ranked alphabet Σ of operators or

basic function symbols, a finite ranked alphabet Ω of predicate symbols and a finite set of equations of the form

$$F(x_1 \cdots x_n) = \textit{if } p(\tau_1 \cdots \tau_k) \textit{ then } \alpha \textit{ else } \beta, \qquad (*)$$

exactly one equation for each $F \in \mathscr{F}$. In (*), F is in $\mathscr{F}_n$ ($n \geqslant 0$), p in Ω_k ($k \geqslant 0$), and $\tau_1, \ldots, \tau_k$, α, β are in $T_{\Sigma \cup \mathscr{F}}(X_n)$. An "interpretation" A consists of a set A, for each $f \in \Sigma_k$ a partial function $f_A: A^k \to A$, and for each $p \in \Omega_k$ a partial function $p_A: A^k \to$ {true, false}. For such an interpretation one can, in the usual way (see [18]), associate with each $F \in \mathscr{F}_n$ two partial functions $A^n \to A$ which are the functions computed by (S, F) in A in a call by value or call by name mode.

We now associate with each recursive program scheme S a context-free tree grammar G_S. Define new ranked alphabets Ω^Y and Ω^N such that, for each k, ${\Omega_k}^Y = \{p_Y \mid p \in \Omega_k\}$ and ${\Omega_k}^N = \{p_N \mid p \in \Omega_k\}$, and let pr_2 be a new symbol of rank 2. G_S is defined to be $(\Sigma \cup \Omega^Y \cup \Omega^N \cup \{pr_2\}, \mathscr{F}, R)$, where, corresponding to each equation $F(x_1 \cdots x_n) = \textit{if } p(\tau_1 \cdots \tau_k) \textit{ then } \alpha \textit{ else } \beta$, the set R contains the two rules

$$F(x_1 \cdots x_n) \to pr_2(p_Y(\tau_1 \cdots \tau_k)\alpha)$$

and

$$F(x_1 \cdots x_n) \to pr_2(p_N(\tau_1 \cdots \tau_k)\beta).$$

Finally we associate with each interpretation A a partial $(\Sigma \cup \Omega^Y \cup \Omega^N \cup \{pr_2\})$-algebra A' with the same carrier, for $f \in \Sigma_k$, $f_{A'} = f_A$, for $p \in \Omega_k$

$$p_{YA'} = \{(a_1, \ldots, a_k, a_1) \mid p_A(a_1, \ldots, a_k) = \text{true}\}$$

and

$$p_{NA'} = \{(a_1, \ldots, a_k, a_1) \mid p_A(a_1, \ldots, a_k) = \text{false}\},$$

and $pr_{2A'} = \{(a_1, a_2, a_2) \mid a_1, a_2 \in A\}$.

We now state without proof the validness of these translations. For any recursive program scheme S, function symbol F and interpretation A,

(1) the partial function computed by (S, F) in A in a call by value mode is equal to the component of F in the solution of G_S in $\mathscr{R}(A')$ (i.e., the call by value relation computed by (G_S, F) over A'; see the comment following Corollary 5.7);

(2) the partial function computed by (S, F) in A in a call by name mode is equal to the restriction to singletons of the component of F in the solution of G_S in $\mathscr{F}_\Delta(\mathscr{P}(A'))$ (i.e., the call by name relation computed by (G_S, F) over A'; see Definition 5.19).

In [18], to obtain the call by name function computed by a recursive program scheme for an interpretation A, the domain A is extended to $A \cup \{\omega\}$, where ω is a new element standing for "undefined." It is then shown that a recursive program scheme defines a total function $(A \cup \{\omega\})^n \to A \cup \{\omega\}$, which is the least fixed point of a suitable mapping. In our formalism, the call by name relation is obtained via a total function $(\mathscr{P}(A))^n \to \mathscr{P}(A)$. Obviously, in $\mathscr{P}(A)$, the empty set plays the role of ω. We state without proof:

(3) The total function computed by (S, F), in a call by name mode, in $A \cup \{\omega\}$ is equal to the restriction to singletons and ϕ (where ϕ stands for ω) of the F-component of the solution of G_S in $\mathscr{F}_\Delta(\mathscr{P}(A'))$.

From correspondences (1) and (2), and Theorems 5.6 and 5.17, respectively, we obtain the following MW-like results for recursive program schemes. For any recursive program scheme S, function symbol F and interpretation A,

(1) the call by value function computed by (S, F) in A is the D-homomorphic image in $\mathscr{R}(A')$ of a recognizable tree language over $D = D(\tilde{\Sigma})$, where $\tilde{\Sigma} = \Sigma \cup \Omega^Y \cup \Omega^N \cup \{pr_2\}$ (note that we may delete pr_2 from $\tilde{\Sigma}$ since its homomorphic image is equal to that of π_2^2);

(2) the call by name function computed by (S, F) in A is the derived relation over A' of an OI tree language with variables over the alphabet $\Sigma \cup \Omega^Y \cup \Omega^N \cup \{pr_2\}$ (see the comment following Definition 5.19).

What can we say about recursive program schemes more general than the ones discussed above? Obviously, we can also handle nondeterminism. Consider, for instance, the following program taken from [38]): $F(x) = \textit{if}\ \text{prime}(x)\ \textit{then}\ (x\ \textit{or}\ F(x+1))\ \textit{else}\ F(x+1)$. This nondeterministic program computes (both in the call by value and call by name cases) the relation $\{(x, y) \mid y \text{ is prime and } y \geqslant x\}$ over $\mathbb{N}$. Let $f(x)$ stand for $x + 1$ and $m(x)$ for prime(x). Then a context-free tree grammar computing the same relation has rules

$$F(x) \to pr_2(m_Y(x)x),$$
$$F(x) \to pr_2(m_Y(x)F(f(x))),$$

and

$$F(x) \to pr_2(m_N(x)F(f(x)))$$

or equivalently

$$F(x) \to pr_2(m_Y(x)x)$$

and

$$F(x) \to F(f(x)).$$

What about the basic operations? In the presence of tests it does not seem to make much sense to make them multivalued. However, in the call by name case it makes some sense to consider basic operations $(A \cup \{\omega\})^n \to A \cup \{\omega\}$, or in other words $\mathscr{P}(A)^n \to \mathscr{P}(A)$, which are Δ-continuous but not $\sqcup$-continuous in their arguments (i.e., they cannot be obtained from A by the subset construction, or in the words of [18] they are not "natural extensions" of partial functions $A^n \to A$). We have seen previously that our MW-like result breaks down for non $\sqcup$-continuous algebras. It is, however, not clear whether such operations are needed in the presence of nondeterminism. In [18] the main such operation is *if–then–else*: $(A \cup \{\omega\})^3 \to A \cup \{\omega\}$. We have seen how to handle

if–then–else with the use of nondeterminism. Another example in [18] is the so-called "parallel multiplication" $*: (\mathbb{N} \cup \{\omega\})^2 \to \mathbb{N} \cup \{\omega\}$ where

$$\begin{aligned} *(x, y) &= 0 && \text{if } x = 0 \text{ or } y = 0 \\ &= \omega && \text{if } x = \omega \text{ or } y = \omega \text{ (and } x \neq 0, y \neq 0) \\ &= xy && \text{otherwise (where } xy \text{ is the ordinary product} \\ & && \text{of natural numbers } x \text{ and } y). \end{aligned}$$

But, using nondeterminism, parallel multiplication can actually be programmed, as follows

$$*(xy) = (\textit{if } x = 0 \textit{ then } 0 \textit{ else } xy) \textit{ or } (\textit{if } y = 0 \textit{ then } 0 \textit{ else } xy),$$

where the call by name solution is of course intended. Thus it seems that, if such functions are needed, they can as well be programmed rather than considered to be basic.

We conclude this section by connecting our MW-like results to MW-like results obtained in the literature in connection with *infinite trees* (see [15, 22, 23]).

In [15] it is shown that a system of regular Σ-equations in which, for each nonterminal, the right-hand side of its equation is a singleton (let us call this a "deterministic" system), can be solved in the algebra CT_Σ of infinite trees over Σ (we shall call its solution a "recognizable infinite tree"). Let $\perp$ be a new symbol of rank 0. Then CT_Σ can be viewed as consisting of all $\Sigma \cup \{\perp\}$-labeled finite or infinite trees such that a node labeled with a symbol of rank n has exactly n successors. The finite trees in CT_Σ are therefore those of $T_\Sigma(\perp)$. A natural order, with minimal element $\perp$, is defined on CT_Σ, and it is shown that CT_Σ is free in the class of all Δ-continuous Σ-algebras with Δ-continuous $\perp$-preserving Σ-homomorphisms. Moreover, the solution of a deterministic system of regular Σ-equations in any Δ-continuous Σ-algebra is the homomorphic image of its solution in CT_Σ. A deterministic system of context-free Σ-equations (i.e., a context-free grammar with one rule for each nonterminal) can be solved in $CT_\Sigma(X)$ (we shall call its solution a "context-free infinite tree"). In fact, $CT_\Sigma(X)$ is a (Δ-continuous) $D(\Sigma)$-algebra in the obvious way (with $c_{n,k}$ being substitution of infinite trees). For any Δ-continuous Σ-algebra B such a system of context-free Σ-equations can be solved in $\mathscr{F}_\Delta(B)$ and the solution is the derived operation of the solution in $CT_\Sigma(X)$ (where "derived operation" is the unique Δ-continuous $\perp$-preserving $D(\Sigma)$-homomorphism from $CT_\Sigma(X)$ into $\mathscr{F}_\Delta(B)$). For more details we refer to [15]. In a different setting these MW-like results have been shown by Nivat [23], who represents an infinite tree by a Δ-complete subset of $T_\Sigma(\perp)$.

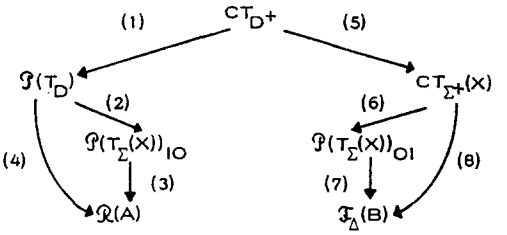
Obviously, these results can also be applied to "nondeterministic" systems of equations by the introduction of an operator $+$ to represent union. For context-free equations there are two (equivalent) ways of doing this, depending on whether $+$ is viewed as an operator on the Σ level or on the $D(\Sigma)$ level. We shall consider the latter alternative, leaving the former to the reader.

Let, for any S-sorted alphabet Ω, Ω^+ denote the S-sorted alphabet consisting of $\Omega \cup \{+_s \mid s \in S\}$, where $+_s$ is a new symbol of type $\langle ss, s\rangle$. Whenever s is understood we write $+$ rather than $+_s$. With each system E of regular Ω-equations we now associate

a deterministic system E^+ of regular Ω^+-equations as follows. If $F = \{\tau_1, ..., \tau_n\}$ is an equation of E (with $n \geqslant 1$), then $F = \{\tau_1 + \tau_2 + \cdots + \tau_n\}$ (in some arbitrary order) is the equation for F in E^+, where $\tau_1 + \cdots + \tau_n$ of course stands for $+_s(\tau_1 +_s (\tau_2 + \cdots +_s (\tau_{n-1}\tau_n) \cdots))$, s being the sort of F. If $F = \varnothing$ is an equation of E, then $F = \{F\}$ is the equation for F in E^+. It should be obvious that, for any Δ-continuous Ω-algebra A with $\sqcup$-complete carriers, the solutions of E and E^+ are equal, if, for each s in S, $+_s$ is interpreted as the $\sqcup$-operation in A_s. However, E^+ can also be solved in Δ-continuous Ω^+-algebras in which $+$ is not interpreted as $\sqcup$, for example in CT_{Ω^+}.

Let now Σ be a ranked alphabet and D its derived alphabet. Note that $\Sigma^+ = \Sigma \cup \{+\}$ with $+ \in \Sigma_2^+$ and $D^+ = D \cup \{+_n \mid n \in \mathbb{N}\}$ with $+_n \in D^+_{nn,n}$. For a given system E of regular D-equations, we can solve E^+ in the D^+-algebras $\mathscr{P}(T_\Sigma(X))_{IO}$, $\mathscr{P}(T_\Sigma(X))_{OI}$, $\mathscr{R}(A)$, and $\mathscr{F}_\Delta(B)$ (where A is a nondeterministic Σ-algebra and B is a Δ-continuous Σ-algebra with $\sqcup$-complete carrier) as before, but also in the Δ-continuous D^+-algebras $CT_{\Sigma^+}(X)$ and CT_{D^+} (where, for $t_1, t_2 \in CT_{\Sigma^+}(X_n)$, $+_n(t_1, t_2) = +(t_1 t_2)$).

We now obtain the following diagram



In this diagram, if two algebras P and Q are connected by arrows, it means that there is a unique Δ-continuous $\perp$-preserving (and whenever possible $\sqcup$-continuous) D^+-homomorphism from P to Q (details are left to the reader). Furthermore, for any deterministic system of regular D^+-equations, its solution in Q is the homomorphic image of its solution in P. We note that one can easily associate with each deterministic system E^+ of regular D^+-equations a system E of regular D-equations such that the solution of E^+ in any Δ-continuous D^+-algebra is a subvector of the solution of E. Hence, as an example, the IO tree languages over Σ are precisely the images under the homomorphism (2) $\circ$ (1) of the recognizable infinite trees in CT_{D^+}. Thus the diagram surveys all MW-like results for systems of regular D-equations discussed up until now.

Although the diagram is meant to be transitive, (4) and (8) are drawn separately because (3) and (7) do not always exist. In fact, as shown previously, (3) only exists if A is a (deterministic) Σ-algebra, whereas (7) only exists if B is a $\sqcup$-continuous Σ-algebra.

Let us recall some of the names of the homomorphisms in the diagram and give names to the new ones. (2) is YIELD, (3) is derrel_A, and (7) is derop_B. For any many-sorted alphabet Ω we shall denote by SET the unique Δ-continuous $\perp$-preserving Ω^+-homomorphism $CT_{\Omega^+} \to \mathscr{P}(T_\Omega)$. Thus (1) in the diagram is SET. It is also appropriate to denote (6) by SET. For obvious reasons we shall denote by YIELD the homomorphism (5): $CT_{D^+} \to CT_{\Sigma^+}(X)$. Note that the homomorphism from CT_{D^+} into $\mathscr{P}(T_\Sigma(X))_{IO}$ is

YIELD $\circ$ SET, whereas the homomorphism from CT_{D+} into $\mathscr{P}(T_\Sigma(X))_{OI}$ is SET $\circ$ YIELD. This expresses in a nice way the basic difference between IO and OI.

Consider a domain of computation A and let $B = \mathscr{P}(A)$. For nondeterministic recursive program schemes we now indicate the "best MW-like result," i.e., the lowest tree algebra in the diagram in which the computation can be done symbolically. There are six possibilities, depending on whether the computation is call by value (IO) or call by name (OI) and depending on whether A is an ordinary Σ-algebra, a nondeterministic Σ-algebra, or has even nonnaturally extended basic operations.

IO	Σ-algebra	$\mathscr{P}(T_\Sigma(X))_{IO}$
	nondet. Σ-algebra	$\mathscr{P}(T_D)$
	nonnat. ext.	meaningless
OI	Σ-algebra	$\mathscr{P}(T_\Sigma(X))_{OI}$
	nondet. Σ-algebra	$\mathscr{P}(T_\Sigma(X))_{OI}$
	nonnat. ext.	$CT_{\Sigma+}(X)$

We finally observe that, instead of $+$, one can also consider a ternary operation *if–then–else*. This would give a similar diagram for deterministic recursive program schemes.

6. A Closure Property of the IO Tree Languages

In Section 4 (Corollary 4.12) we have shown that every IO tree language over Σ is the YIELD of a recognizable tree language over $D(\Sigma)$ and vice versa. This result can be used to prove properties of IO tree languages by applying well-known facts from the theory of recognizable tree languages, in the same way as was done for context-free string languages in [30, 35]. Since each IO string language [12] is obviously the yield of an IO tree language, one can even obtain properties of IO string languages from those of recognizable tree languages (cf. Section 4).

To illustrate the fruitfulness of the algebraic fixed point approach to language theory we shall in this section use Corollary 4.12 to give an algebraic proof of the intuitively obvious fact that the IO tree languages are closed under deterministic bottom-up tree transducer mappings. Before doing so we shall look at two special cases: closure under intersection with a recognizable tree language and closure under "tree homomorphisms." At the end of the section two examples are given which show the nonclosure of the IO tree languages under (nondeterministic) relabeling and the nonclosure of the OI tree languages under tree homomorphisms, respectively.

Let us first recall from Section 2.2 that, for any many-sorted alphabet Σ, a tree language over Σ is recognizable iff it is recognizable over a finite subalphabet of Σ. We shall also use, without mentioning, the fact that, if $L \subseteq T_{\Sigma,s}$, then it is recognizable in T_Σ iff it is recognizable in $T_{\bar{\Sigma}}$, where $\bar{\Sigma}$ is the ranked alphabet with $\bar{\Sigma}_n = \bigcup_{w,s} \{\Sigma_{w,s} \mid \lg(w) = n\}$. We shall refer to $\bar{\Sigma}$ as "Σ, viewed as a ranked alphabet."

In order to obtain closure of the IO tree languages under intersection with a recognizable tree language, we prove the next lemma, which is a straightforward generalization of one of Rounds [30] for regular languages.

6.1. LEMMA. *Let Σ be a finite ranked alphabet. If R is a recognizable tree language over Σ, then, for any finite subalphabet D' of $D(\Sigma)$, $\mathrm{YIELD}^{-1}(R) \cap T_{D',0}$ is recognizable.*

Proof. Let Q be the finite Σ-algebra such that $R = h_Q^{-1}(F)$ for some $F \subseteq Q$, where h_Q is the Σ-homomorphism $T_\Sigma \to Q$. Consider now the finite $D(\Sigma)$-algebra $\mathscr{F}(Q)$ (this algebra, or rather der$\mathscr{F}(Q)$, might be called "the algebra of state transition functions of the finite tree automaton Q"). By Section 2.2 the mapping derop_Q is a $D(\Sigma)$-homomorphism $DT_\Sigma(X) \to \mathscr{F}(Q)$. Therefore $\mathrm{derop}_Q \circ \mathrm{YIELD}$ is the unique $D(\Sigma)$-homomorphism from $T_{D(\Sigma)}$ into $\mathscr{F}(Q)$. Denote this homomorphism by g. Note that, for a tree t of sort 0, $\mathrm{derop}_Q(t) = h_Q(t)$. Hence, for sort 0, $g_0^{-1}(F) = \mathrm{YIELD}^{-1}(\mathrm{derop}_Q^{-1}(F)) = \mathrm{YIELD}^{-1}(h_Q^{-1}(F)) = \mathrm{YIELD}^{-1}(R)$. Finally, the restriction of g to $T_{D'}$ is the unique D'-homomorphism from $T_{D'}$ into $\mathscr{F}(Q)$, and the inverse image of F under this mapping is $\mathrm{YIELD}^{-1}(R) \cap T_{D',0}$. Consequently, this set is recognizable. ∎

6.2. COROLLARY. *The class of* IO *tree languages is closed under intersection with a recognizable tree language.*

Proof. Let L be an IO tree language over Σ and R a recognizable tree language over Σ. By Corollary 4.12, $L = \mathrm{YIELD}(R_0)$ for some recognizable tree language R_0 in $T_{D',0}$, where D' is a finite subalphabet of $D(\Sigma)$. Now $L \cap R = \mathrm{YIELD}(R_0 \cap \mathrm{YIELD}^{-1}(R))$. By the previous lemma $\mathrm{YIELD}^{-1}(R) \cap T_{D',0}$ is recognizable, and so, since the class of recognizable tree languages is closed under intersection, $R_0 \cap \mathrm{YIELD}^{-1}(R)$ is recognizable. Hence, by Corollary 4.12, $\mathrm{YIELD}(R_0 \cap \mathrm{YIELD}^{-1}(R))$ is an IO tree language. ∎

Note that it follows easily from this corollary and the lemma in [30, p. 110] that the class of IO string languages is closed under intersection with a regular string language.

Next we shall show that the class of IO tree languages is closed under "tree homomorphisms." Let Σ and Ω be possibly infinite ranked alphabets. Consider a family $h = \langle h_n \rangle_{n \in \mathbb{N}}$ of mappings $h_n: \Sigma_n \to T_\Omega(X_n)$. Such a family determines a mapping $\bar{h}: T_\Sigma \to T_\Omega$, called a *tree homomorphism*, by the requirements

(i) for $f \in \Sigma_0$, $\bar{h}(f) = h_0(f)$;

(ii) for $f \in \Sigma_n$, $\bar{h}(f(t_1 \cdots t_n)) = h_n(f)[\bar{h}(t_1), \ldots, \bar{h}(t_n)]$.

Moreover, together with the requirement that $\bar{h}(x_i) = x_i$ for all i, $\bar{h}$ is a mapping from $T_\Sigma(X_n)$ into $T_\Omega(X_n)$ for each $n \geqslant 0$. Thus $\bar{h}$ may be viewed as a mapping $DT_\Sigma(X) \to DT_\Omega(X)$. Let $\tilde{D}$ be the $\mathbb{N}$-sorted alphabet $D(\Sigma) - \Sigma'$ (thus $\tilde{D} = D(\Omega) - \Omega'$; $\tilde{D}$ consists of all projection symbols and composition symbols). It can easily be shown, and in fact it is a special case of Lemma 3.3(1), that $\bar{h}$ is a $\tilde{D}$-homomorphism from $DT_\Sigma(X)$ into $DT_\Omega(X)$, both viewed as $\tilde{D}$-algebras.

A tree homomorphism $\bar{h}$ is called *linear* if no $h_n(f)$ contains two occurrences of the same variable. In the next lemma we show that each tree homomorphism from T_Σ into T_Ω can

be simulated "on the second level" (i.e., on the level of $T_{D(\Sigma)}$ and $T_{D(\Omega)}$) by a linear tree homomorphism.

6.3. LEMMA. *Let Σ and Ω be ranked alphabets, and $\bar{h}$ a tree homomorphism from T_Σ into T_Ω. Then there is a linear tree homomorphism $\bar{g}$: $T_{\overline{D(\Sigma)}} \to T_{\overline{D(\Omega)}}$ (i.e., $D(\Sigma)$ and $D(\Omega)$ are viewed as ranked alphabets), such that* YIELD $\circ\, \bar{g} = \bar{h} \circ$ YIELD. *Moreover, $\bar{g}$ is sort preserving (for every t, $\bar{g}(t)$ has the same sort as t).*

Proof. Intuitively, in order to simulate $\bar{h}$, $\bar{g}$ only has to change the frontiers of the $D(\Sigma)$-trees. Formally $\bar{g}$ is defined as follows (note that $D(\Sigma)$ and $D(\Omega)$ are viewed as ranked alphabets):

(i) for $n \geqslant 0$ and $f \in \Sigma_n$, $g_0(f') = \mathrm{COMB}_n^{\Omega}(h_n(f))$ (for the definition of COMB, see Definition 4.4);

(ii) for $1 \leqslant i \leqslant n$, $g_0(\pi_i^n) = \pi_i^n$;

(iii) for $n, k \geqslant 0$, $g_{n+1}(c_{n,k}) = c_{n,k}(x_1 \cdots x_{n+1})$.

Clearly, $\bar{g}$ may be viewed as a $\tilde{D}$-homomorphism from $T_{D(\Sigma)}$ into $T_{D(\Omega)}$, both considered as $\tilde{D}$-algebras (where $\tilde{D} = D(\Sigma) - \Sigma'$). Now we have the following diagram

$$\begin{array}{ccc} T_{D(\Sigma)} & \xrightarrow{\bar{g}} & T_{D(\Omega)} \\ \mathrm{YIELD}\downarrow & & \downarrow\mathrm{YIELD} \\ DT_\Sigma(X) & \xrightarrow{\bar{h}} & DT_\Omega(X) \end{array}$$

where all sets are $\tilde{D}$-algebras and all mappings are $\tilde{D}$-homomorphisms. Since $T_{D(\Sigma)}$ is obviously the free $\tilde{D}$-algebra generated by the elements of Σ', the diagram commutes if it does for the generators. For $n \geqslant 0$ and $f \in \Sigma_n$,

$$\mathrm{YIELD}(\bar{g}(f')) = \mathrm{YIELD}(\mathrm{COMB}_n^{\Omega}(h_n(f))) = h_n(f) = \bar{h}(f(x_1 \cdots x_n)) = \bar{h}(\mathrm{YIELD}(f')).$$

Hence YIELD $\circ\, \bar{g} = \bar{h} \circ$ YIELD. ∎

6.4. COROLLARY. *The class of* IO *tree languages is closed under tree homomorphisms.*

Proof. Let L be an IO tree language over the finite ranked alphabet Σ. Thus $L =$ YIELD(R) for some recognizable tree language R over some finite subalphabet of $D(\Sigma)$. By the previous lemma, for any tree homomorphism $\bar{h}$: $T_\Sigma \to T_\Omega$, $\bar{h}(L) = \bar{h}(\mathrm{YIELD}(R)) =$ YIELD$(\bar{g}(R))$, where $\bar{g}$ is a linear tree homomorphism $T_{D(\Sigma)} \to T_{D(\Omega)}$, which may be restricted to the above mentioned finite subalphabet of $D(\Sigma)$. Therefore the closure of the recognizable tree languages under linear tree homomorphisms [34] implies that $\bar{g}(R)$ is recognizable over $D(\Omega)$. Hence YIELD$(\bar{g}(R))$ is an IO tree language. ∎

We now turn to the slightly more complicated case of a deterministic bottom-up tree transducer, which may be treated by combining the previous two lemmas. We shall show that such a tree transducer may be simulated "on the second level" by a linear (nondeterministic) bottom-up tree transducer. Using the closure of the recognizable tree

languages under linear tree transducers, we obtain the desired closure result as in the two previous cases. For background on bottom-up tree transducers, see [2, 9].

Let Σ and Ω be finite ranked alphabets. A *bottom-up tree transducer B from Σ to Ω* (called a "bottom-up finite state transformation" in [9]) consists of a finite set Q of "states," a subset F of Q (of "final" states) and a family $\langle B_n\rangle_{n\in\mathbb{N}}$ of mappings B_n: $\Sigma_n \to \mathscr{P}(Q^n \times Q \times T_\Omega(X_n))$, such that, for each $f\in\Sigma_n$, $B_n(f)$ is finite.

Intuitively a tuple $(q_1,\ldots, q_n, q, s)$ in $B_n(f)$ corresponds to a rule $f(q_1(x_1)\cdots q_n(x_n)) \to q(s)$, see [9]. We shall be careless with parentheses, thus $(a,(b,c)) = ((a,b),c) = (a,b,c)$, etc.

B is called *deterministic* if, for all $n\geqslant 0$ and $f\in\Sigma_n$, $B_n(f)$ is in fact a mapping $Q^n \to Q \times T_\Omega(X_n)$ (in particular, $B_0(f)$ is a single element). B is called *linear* if all trees from $T_\Omega(X)$ used in $\langle B_n\rangle_{n\in\mathbb{N}}$ are linear (i.e., each variable occurs at most once in the tree).

B determines a family $\bar{B} = \langle \bar{B}_n\rangle_{n\in\mathbb{N}}$ of mappings $\bar{B}_n$: $T_\Sigma(X_n) \to \mathscr{P}(Q^n \times Q \times T_\Omega(X_n))$ as follows (intuitively, $(q_1,\ldots, q_n, q, s)\in \bar{B}_n(t)$ iff $t[q_1(x_1),\ldots, q_n(x_n)] \overset{*}{\Rightarrow} q(s)$, i.e., when started on t in state q_i at each occurrence of x_i, B can arrive in state q with output s, see [2, 9]):

(i) for $1\leqslant i\leqslant n$, $\bar{B}_n(x_i) = \{(q_1,\ldots, q_n, q_i, x_i) \mid q_1,\ldots, q_n\in Q\}$;

(ii) for $f\in\Sigma_0$, $\bar{B}_n(f) = \{(q_1,\ldots, q_n, q, s) \mid q_i\in Q$ and $(q,s)\in B_0(f)\}$;

(iii) for $f\in\Sigma_k$ and $t_1,\ldots, t_k\in T_\Sigma(X_n)$, $\bar{B}_n(f(t_1\cdots t_k)) = \{(q_1,\ldots, q_n, q, s) \mid$ there exist $p_1,\ldots, p_k$ in Q and $u_1,\ldots, u_k$, u in $T_\Omega(X_n)$ such that, for $1\leqslant i\leqslant k$, $(q_1,\ldots, q_n, p_i, u_i)\in \bar{B}_n(t_i)$, $(p_1,\ldots, p_k, q, u)\in B_k(f)$ and $s = u[u_1,\ldots, u_k]\}$.

In particular, $\bar{B}_0$ is a mapping from T_Σ into $\mathscr{P}(Q \times T_\Omega)$.

B realizes a mapping from T_Σ into $\mathscr{P}(T_\Omega)$, also denoted by B, defined by $B(t) = \{s \mid (q,s)\in \bar{B}_0(t)$ for some q in $F\}$. Moreover, for $L\subseteq T_\Sigma$, we define $B(L) = \bigcup_{t\in L} B(t)$. Note that, for deterministic transducers, B is a partial function $T_\Sigma \to T_\Omega$.

We now show that every deterministic bottom-up tree transducer can be "lifted to the second level."

6.5. LEMMA. *Let Σ and Ω be finite ranked alphabets, and B a deterministic bottom-up tree transducer from Σ into Ω. For every finite subalphabet D' of $D(\Sigma)$ there is a linear bottom-up tree transducer U' from D' into $D(\Omega)$ (both viewed as ranked alphabets) such that* YIELD $\circ\, U' = B \circ$ YIELD *as mappings $T_{D',0} \to \mathscr{P}(T_\Omega)$ (in fact they are partial functions $T_{D',0} \to T_\Omega$).*

Proof. We shall construct an infinite linear bottom-up tree transducer U from $D(\Sigma)$ into $D(\Omega)$ such that YIELD $\circ\, U = B \circ$ YIELD as mappings $T_{D(\Sigma),0} \to \mathscr{P}(T_\Omega)$. U will have an infinite number of states and an infinite number of "input symbols" (the elements of $D(\Sigma)$), but otherwise all previously given definitions also apply to U. It will be left to the reader to see that, for any finite subalphabet D' of $D(\Sigma)$, U can be restricted in an obvious way to an ordinary bottom-up tree transducer U' with the required property.

Intuitively, U will be constructed in such a way that, if YIELD$(s) = t$, then U simulates on s the behavior of B on t by guessing for each occurrence of an $f'\in\Sigma_n'$ at the frontier of s what the rule applied by B at each of the corresponding occurrences

of f in t will be, and then checking the consistency of these guesses (note that, since B is deterministic, it will apply the same rule at two occurrences of f in t which correspond to the same occurrence of f' in s). The states of U will therefore be the state transitions of B. U changes the $D(\Sigma)$-trees only at their frontier.

Formally U is defined as follows. Let B have states Q, final states F, and mappings $\langle B_n \rangle_{n\in\mathbb{N}}$. Then the set of states of U is $Q_U = \bigcup_{n\in\mathbb{N}} (Q^n \times Q)$, and the set of final states of U is F. The mappings $\langle U_n \rangle_{n\in\mathbb{N}}$ are defined by

(i) for $f \in \Sigma_n$ $(n \geqslant 0)$,

$$U_0(f') = \{((q_1, \ldots, q_n, q), \mathrm{COMB}_n^{\Omega}(t)) \mid (q_1, \ldots, q_n, q, t) \in B_n(f)\};$$

(ii) for $1 \leqslant i \leqslant n$, $U_0(\pi_i^n) = \{((q_1, \ldots, q_n, q_i), \pi_i^n) \mid q_1, \ldots, q_n \in Q\}$;

(iii) for $n, k \geqslant 0$,

$$U_{n+1}(c_{n,k}) = \{((q_1, \ldots, q_k, p_1), \ldots, (q_1, \ldots, q_k, p_n), (p_1, \ldots, p_n, p), (q_1, \ldots, q_k, p), c_{n,k}(x_1 \cdots x_{n+1})) \mid p_i, p_j, p \in Q\}.$$

Consider the mapping $\bar{U}_0$: $T_{D(\Sigma)} \to \mathscr{P}(Q_U \times T_{D(\Omega)})$. It is easy to see that, for each $n \geqslant 0$, $\bar{U}_0$ maps $T_{D(\Sigma),n}$ into $\mathscr{P}(Q^n \times Q \times T_{D(\Omega),n})$. Thus one can draw the following diagram

$$\begin{array}{ccc} T_{D(\Sigma),n} & \xrightarrow{\bar{U}_0} & \mathscr{P}(Q^n \times Q \times T_{D(\Omega),n}) \\ {\scriptstyle \mathrm{YIELD}}\downarrow & & \downarrow{\scriptstyle h} \\ T_{\Sigma}(X_n) & \xrightarrow[\bar{B}_n]{} & \mathscr{P}(Q^n \times Q \times T_{\Omega}(X_n)) \end{array}$$

where h transforms each element of $T_{D(\Omega),n}$ into YIELD in $T_{\Omega}(X_n)$ (and perhaps "removes some parentheses"). Clearly, to prove our lemma, it suffices to show the commutativity of the above diagram for all n (in particular $n = 0$). We shall do this, analogously to the case of tree homomorphisms, by finding four $\tilde{D}$-algebras such that the sets in the above diagram are their carriers of sort n, and such that the mappings in the diagram become $\tilde{D}$-homomorphisms (where, as before, $\tilde{D} = D(\Sigma) - \Sigma' = D(\Omega) - \Omega'$). For the left side of the diagram we can choose the $\tilde{D}$-algebras $T_{D(\Sigma)}$ and $DT_{\Sigma}(X)$. For the right side of the diagram, let $TUP(Q)$ denote the (partial) $\tilde{D}$-algebra such that $Q^n \times Q$ is the domain of sort n, $\pi_i^n = \{(q_1, \ldots, q_n, q_i) \mid q_1, \ldots, q_n \in Q\}$, and $c_{n,k}((q_1, \ldots, q_k, p_1), \ldots, (q_1, \ldots, q_k, p_n), (p_1, \ldots, p_n, p)) = (q_1, \ldots, q_k, p)$ and undefined otherwise. Then the $\tilde{D}$-algebras $\mathscr{P}(TUP(Q) \times T_{D(\Omega)})$ and $\mathscr{P}(TUP(Q) \times DT_{\Omega}(X))$ have the proper domains (here, $\times$ denotes the obvious products of algebras, and $\mathscr{P}$ the subset algebra operation). Our diagram is now transformed into

$$\begin{array}{ccc} T_{D(\Sigma)} & \xrightarrow{\bar{U}_0} & \mathscr{P}(TUP(Q) \times T_{D(\Omega)}) \\ {\scriptstyle \mathrm{YIELD}}\downarrow & & \downarrow{\scriptstyle h} \\ DT_{\Sigma}(X) & \xrightarrow[\bar{B}]{} & \mathscr{P}(TUP(Q) \times DT_{\Omega}(X)) \end{array}$$

Obviously, YIELD and h are both $\tilde{D}$-homomorphisms. It follows easily from the definition of U that $\bar{U}_0$ is a $\tilde{D}$-homomorphism. Finally, it can be shown that $\bar{B}$ is also a $\tilde{D}$-homomorphism (here the determinism of B is essential; the proof is similar to that of Lemma 3.3(2)). We leave it to the reader to check the details. Now, $T_{D(\Sigma)}$ is the free $\tilde{D}$-algebra generated by Σ'. But, for $f \in \Sigma_n$, $h(\bar{U}_0(f')) = h(U_0(f')) = \{(q_1, \ldots, q_n, \text{YIELD}(\text{COMB}_n^Q(t))) \mid (q_1, \ldots, q_n, q, t) \in B_n(f)\} = B_n(f) = \bar{B}_n(f(x_1 \cdots x_n)) = \bar{B}_n(\text{YIELD}(f'))$. Hence $h \circ \bar{U}_0 = \bar{B} \circ \text{YIELD}$, and the lemma is proved. ∎

As a corollary we obtain the main theorem of this section.

6.6. THEOREM. *The class of* IO *tree languages is closed under deterministic bottom-up tree transducer mappings.*

Proof. Let L be an IO tree language over Σ. Thus $L = \text{YIELD}(R)$ for some recognizable R over a finite subalphabet D' of $D(\Sigma)$. By the previous lemma, for any deterministic bottom-up tree transducer B, $B(L) = B(\text{YIELD}(R)) = \text{YIELD}(U'(R))$ where U' is a linear bottom-up tree transducer from D' to $D(\Omega)$. Since the class of recognizable tree languages is closed under linear tree transducer mappings [9, 34], $U'(R)$ is recognizable over $D(\Omega)$, and hence its YIELD is an IO tree language. ∎

Rounds [29, 30] has shown that the class of OI tree languages is closed under linear top-down tree transducer mappings (and hence under linear bottom-up tree transducer mappings, see [9]). This closure result and that of Theorem 6.6 are optimal in the sense that the OI tree languages are not closed under copying (more precisely, under tree homomorphisms), whereas the IO tree languages are not closed under nondeterminism (more precisely, under nondeterministic relabeling). This can easily be shown from the examples given by Fischer to show the incomparability of the classes of IO and OI string languages (for a definition of these string languages, see Definition 7.8 or [12]).

6.7. EXAMPLE. (The OI tree languages are not closed under tree homomorphisms.) The string language $L = \{b^m(ab^m)^{n-1} \mid m \geqslant 1, n = 2^m\}$ is not an OI string language [12]. Let $G = (\Sigma, \mathscr{F}, P)$ be the OI tree grammar with $\Sigma_0 = \{a, b\}$, $\Sigma_1 = \{g\}$, $\Sigma_2 = \{c\}$, $\mathscr{F}_0 = \{S\}$, $\mathscr{F}_1 = \{F\}$ and P consists of the rules

$$S \to F(b),$$
$$F(x_1) \to g(F(c(x_1 b))),$$

and

$$F(x_1) \to g(x_1).$$

Let h be the tree homomorphism with $h_1(g) = c(x_1 c(a x_1))$ and the identity on the other symbols (i.e., $h_0(a) = a$, $h_0(b) = b$, and $h_2(c) = c(x_1 x_2)$). Then $\text{yield}(h(L_{\text{OI}}(G, S))) = L$. Hence $h(L_{\text{OI}}(G, S))$ is not an OI tree language. ∎

6.8. EXAMPLE. (The IO tree languages are not closed under nondeterministic relabeling.)

The string language $L = \{w \in \{a, b\}^* \mid$ the number of symbols b in w is 2^n for some $n \geq 0\}$ is not an IO string language [12]. Consider the IO tree grammar $G = (\Sigma, \mathscr{F}, P)$ with $\Sigma_0 = \{a, b, e\}$, $\Sigma_2 = \{c\}$, $\mathscr{F}_0 = \{S, A\}$, $\mathscr{F}_1 = \{F\}$ and P consists of the rules

$$S \to c(AF(c(bA))),$$
$$A \to c(aA), \qquad A \to e,$$
$$F(x_1) \to F(c(x_1x_1)), \qquad F(x_1) \to x_1 .$$

(G is obtained from the macrogrammar with rules $S \to AF(bA)$, $A \to aA$, $A \to \lambda$, $F(x_1) \to F(x_1x_1)$, $F(x_1) \to x_1$ by replacing λ by e and writing c for concatenation). It is easy to see that $\text{yield}(L_{\text{IO}}(G, S)) = \{a^m(ba^k)^{2^n} \mid m, k, n \geq 0\}$ (note that $\text{yield}(e) = \lambda$). Let h be the nondeterministic relabeling which relabels a by a or e, and leaves the other symbols (b, e, and c) the same. Then $\text{yield}(h(L_{\text{IO}}(G, S))) = L$. Hence $h(L_{\text{IO}}(G, S))$ is not an IO tree language.

We note that the same argument shows that the class of IO string languages is not closed under nondeterministic relabeling: let h' relabel a by a or f, then $h'(\text{yield}(L_{\text{IO}}(G, S)))$ is not an IO string language (the IO string languages are closed under string homomorphisms, in particular the one which sends f into λ). ∎

7. Hierarchies

We have seen in Sections 3, 4, and 5 that any system of context-free Σ-equations over some Σ-algebra A may be replaced by a system of regular $D(\Sigma)$-equations over some appropriate $D(\Sigma)$-algebra connected to A. Several authors [17, 37, 42] have suggested-that this process may be iterated, i.e., one may consider systems of context-free $D(\Sigma)$ equations which may then be replaced by regular $D(D(\Sigma))$-equations (note that this requires the generalization of "derived alphabet" and other notions to the many-sorted case). In general one may consider systems of regular $D^n(\Sigma)$-equations over an appropriate $D^n(\Sigma)$-algebra A_n corresponding to A. Roughly speaking, A_n consists of "nondeterministic" functions of functions of $\cdots$ of functions (up to level n) over A. In particular each A_n contains the subsets of A. Thus, for growing n, one obtains more and more (at least not less) subsets of A which are solutions of systems of equations, i.e., a hierarchy of higher level equational subsets of A.

In this section we shall show that in fact two such hierarchies can be defined over every Σ-algebra: an IO hierarchy and an OI hierarchy. For both hierarchies an MW-like result can be proved, which, approximately, says that the "level n" equational sets can be obtained by applying the mapping YIELD^n to the recognizable tree languages over $D^n(\Sigma)$ (in the IO case), or the recognizable infinite trees over $D^n(\Sigma)^+$ (in the OI case). In the particular case of a "monadic string algebra" the first three steps in the hierarchy are, in the IO case: the regular, context-free, and IO string languages, and in the OI case: the regular, context-free, and OI string languages. We conjecture that the OI hierarchy is the same as that in [42]. The IO hierarchy is the one suggested in [17].

This section is organized as follows. First we generalize a number of notions to the many-sorted case. We state a theorem saying that, as for recognizable tree languages, no new IO and OI tree languages result from this generalization. Second we define the level n IO equational and level n OI equational subsets of an algebra. We then prove the above mentioned MW-like result for the IO case, and consider the IO string hierarchy. Finally we briefly treat the OI case.

The reader is now asked to generalize most of the concepts and facts treated so far to the case of a many-sorted alphabet. In order to assist him, we shall define a number of these generalized concepts and leave it to the reader to check their properties (note that in Sections 2 and 4 several many-sorted concepts have already been defined; see also [17]).

Let S be a set of sorts. For $w \in S^*$ and $1 \leqslant i \leqslant \lg(w)$, we shall denote by w_i the ith symbol of w, thus $w = w_1 w_2 \cdots w_n$, where $n = \lg(w)$ and $w_1, \ldots, w_n \in S$. Let $\Sigma = \langle \Sigma_{w,s} \rangle$ be an S-sorted alphabet ($\langle w, s \rangle \in S^* \times S$). First of all we need the generalized notion of derived alphabet.

The *derived* $(S^* \times S)$-sorted *alphabet of* Σ, denoted by $D(\Sigma)$, is obtained as follows. Let, for each $\langle w, s \rangle \in S^* \times S$ and each $f \in \Sigma_{w,s}$, f' be a new symbol; let for each $w \in S^*$ $(w \neq \lambda)$ and each i, $1 \leqslant i \leqslant \lg(w)$, $\pi_i{}^w$ be a new symbol; and let for each w, $v \in S^*$ and $s \in S$, $c_{w,v,s}$ be a new symbol. Then $D(\Sigma)$ consists of these new symbols with their types (elements of $(S^* \times S)^* \times (S^* \times S)$) specified as follows:

(i) for $f \in \Sigma_{w,s}$, f' has type $\langle \lambda, \langle w, s \rangle \rangle$;

(ii) $\pi_i{}^w$ has type $\langle \lambda, \langle w, w_i \rangle \rangle$; and

(iii) $c_{w,v,s}$ has type $\langle \langle w, s \rangle \langle v, w_1 \rangle \cdots \langle v, w_n \rangle, \langle v, s \rangle \rangle$ (in particular, $c_{\lambda,v,s}$ has type $\langle \langle \lambda, s \rangle, \langle v, s \rangle \rangle$).

In the ranked case ($S = \{s\}$), to remain consistent with Definition 2.2.1, one has to identify $\pi_i{}^w$ with $\pi_i^{\lg(w)}$, and $c_{w,v,s}$ with $c_{\lg(w),\lg(v)}$.

The *derived alphabet of order* n, denoted by $D^n(\Sigma)$, is defined by $D^0(\Sigma) = \Sigma$ and $D^{n+1}(\Sigma) = D(D^n(\Sigma))$.

In the place of X, we shall use the set of (sorted) *variables* $X_S = \{x_{i,s} \mid i \geqslant 1 \text{ and } s \in S\}$. The symbol $x_{i,s}$ is meant to be a constant of sort s. Let $X_\lambda = \varnothing$ and, for every $w \in S^*$ $(w \neq \lambda)$, $X_w = \{x_{i,w_i} \mid 1 \leqslant i \leqslant \lg(w)\}$. For $w \in S^*$, X_w will also be used to denote the family of disjoint sets $Y = \langle Y_s \rangle_{s \in S}$ where $Y_s = \{x_{i,w_i} \mid w_i = s\}$. Thus $T_\Sigma(X_w)$ denotes $T_\Sigma(Y)$ as defined in Section 2.2. Note that in the ranked case these concepts are the usual ones.

The *tree substitution* $D(\Sigma)$*-algebra*, denoted by $DT_\Sigma(X_S)$, or $DT_\Sigma(X)$ if S is understood, is defined analogously to the ranked case. The domain of sort $\langle w, s \rangle$ is $T_\Sigma(X_w)_s$; for $f \in \Sigma_{w,s}$, f' is the tree $f(x_{1,w_1} \cdots x_{n,w_n})$; $\pi_i{}^w = x_{i,w_i}$ and $c_{w,v,s}(t, t_1, \ldots, t_n) = t[t_1, \ldots, t_n]$, the result of substituting t_i for x_{i,w_i} in t. The unique $D(\Sigma)$-homomorphism $T_{D(\Sigma)} \to DT_\Sigma(X)$ is called YIELD.

Let now A be a Σ-algebra. The $D(\Sigma)$*-algebra of functions over* A, denoted by $\mathscr{F}(A)$, has the set of all functions $A_{s_1} \times \cdots \times A_{s_n} \to A_s$ as domain of sort $\langle s_1 \cdots s_n, s \rangle$ (in particular, $\mathscr{F}(A)_{\langle \lambda, s \rangle} = A_s$); $\pi_i^{s_1 \cdots s_n}$ is the ith projection and $c_{w,v,s}$ is the usual composition of functions. Every $t \in T_\Sigma(X_w)_s$ gives rise to a derived operation t_A in $\mathscr{F}(A)_{\langle w,s \rangle}$ (see Section

2.2). Note that for $w = \lambda$, $t_A = h_A(t)$, where h_A is the Σ-homomorphism $T_\Sigma \to A$. We also denote t_A by $\mathrm{derop}_A(t)$; derop_A is the unique $D(\Sigma)$-homomorphism $DT_\Sigma(X) \to \mathscr{F}(A)$. $\mathscr{F}$ can be iterated and $\mathscr{F}^n(A)$ is clearly a $D^n(\Sigma)$-algebra ($n \geqslant 0$). For a Δ-continuous Σ-algebra B with $\sqcup$-complete carriers, the *$D(\Sigma)$-algebra of Δ-continuous functions over B*, denoted by $\mathscr{F}_\Delta(B)$, is defined in the obvious way. By Lemma 5.14, $\mathscr{F}_\Delta$ can be iterated and $\mathscr{F}_\Delta^n(B)$ is a Δ-continuous $D^n(\Sigma)$-algebra with $\sqcup$-complete carriers.

Finally we define an *S-sorted context-free tree grammar* to be a triple $G = (\Sigma, \mathscr{F}, P)$, where Σ and $\mathscr{F}$ are disjoint finite S-sorted alphabets and P is a finite set of productions of the form $F(x_{1,w_1} \cdots x_{k,w_k}) \to \tau$, where $k \geqslant 0$, $F \in \mathscr{F}_{w,s}$, and $\tau \in T_{\Sigma \cup \mathscr{F}}(X_w)_s$ for $w = w_1 \cdots w_k$ and some s in S. The definitions of derivation and generated language are completely analogous to the ranked case.

This ends our list of generalizations.

We shall first show that in the many sorted case no new IO and OI tree languages are obtained. As before, for any S-sorted alphabet Σ, we shall denote by $\bar{\Sigma}$ the ranked alphabet associated to Σ in a natural way: for $n = 0$, $\bar{\Sigma}_n = \bigcup_{w,s} \{\Sigma_{w,s} \mid \lg(w) = n\}$, that is, $\bar{\Sigma}_n$ consists of all symbols of Σ of rank n.

7.1. THEOREM. *Let Σ be a finite S-sorted alphabet and $\bar{\Sigma}$ its associated (finite) ranked alphabet. Let L be a tree language contained in $T_{\Sigma,s}$ for some $s \in S$. Then L is an* IO (OI) *tree language over Σ if and only if it is an* IO (OI) *tree language over $\bar{\Sigma}$.*

Proof. Let us note first of all that S may be assumed to be finite. The only if part of the statement is easy. One simply changes a given many-sorted context-free tree grammar $G = (\Sigma, \mathscr{F}, P)$ into the ordinary context-free tree grammar $G_1 = (\bar{\Sigma}, \bar{\mathscr{F}}, P_1)$ where P_1 is obtained from P by "dropping the sorts of the variables" (i.e., replacing each $x_{i,s}$ by x_i in all rules). It is easy to see that $L(G_1, A) = L(G, A)$ for any $A \in \bar{\mathscr{F}}_0$ in both the IO and OI mode of derivation.

For the if part, let L be an arbitrary IO (OI) tree language over $\bar{\Sigma}$ (not necessarily contained in $T_{\Sigma,s}$). It suffices to show that $L \cap T_{\Sigma,s}$ is an IO (OI) tree language over Σ. Clearly $T_{\Sigma,s}$ is a recognizable tree language over $\bar{\Sigma}$ (use the set of sorts together with one "rejecting state" as the elements of the obvious finite Σ-algebra recognizing $T_{\Sigma,s}$). Now the IO (OI) tree languages are closed under intersection with a recognizable tree language (for IO, see Corollary 6.2; for OI, see [29, 30]). Thus, in particular, $L \cap T_{\Sigma,s}$ is an IO (OI) tree language over $\bar{\Sigma}$. However, by inspecting the constructions involved in the above mentioned proofs, it can be seen that one ends up with a grammar for $L \cap T_{\Sigma,s}$ which can easily be changed into a many-sorted grammar by associating types with nonterminals. In fact the only problem is deletion: a nonterminal might produce a "nonsorted" subtree, which is deleted later in the derivation. In the IO case this is solved by starting with a nondeleting grammar for L (see [12]), and the OI case by simply not deriving the wrong subtree. ∎

We now turn to the definition of the hierarchies of higher level equational subsets of a Σ-algebra A. First we note that the IO equational subsets of A (see Section 5) can also be characterized as the solutions of systems of regular $D(\Sigma)$-equations in the $D(\Sigma)$-algebra $\mathscr{P}(\mathscr{F}(A))$ rather than in $\mathscr{R}(A)$ (there is a $\sqcup$-continuous $D(\Sigma)$-homomorphism h:

$\mathscr{P}(\mathscr{F}(A)) \to \mathscr{R}(A)$; in fact, for $Q \subseteq \mathscr{F}(A)$, $h(Q) = \sqcup Q$, where $\mathscr{F}(A)$ is considered as a subalgebra of $\mathscr{R}(A)$ in the obvious way: hence, since h is the identity on $\mathscr{P}(A)$, the $D(\Sigma)$-equational elements of $\mathscr{P}(A)$ are the same in $\mathscr{P}(\mathscr{F}(A))$ and $\mathscr{R}(A)$). Moreover, from an intuitive point of view, it is perhaps more natural to solve $D(\Sigma)$-equations in $\mathscr{P}(\mathscr{F}(A))$, where a set of derived operators (i.e., $L \subseteq T_\Sigma(X)$) is interpreted as a set of derived operations, rather than a derived relation.

Consider a derived alphabet $D^n(\Sigma)$ of order n. Intuitively, the composition symbols in $D^n(\Sigma)$ should be interpreted as composition of functions of level n over some domain. Two natural choices of a $D^n(\Sigma)$-algebra connected to a Σ-algebra A are $\mathscr{P}(\mathscr{F}^n(A))$ in the IO case and $\mathscr{F}_\Delta^n(\mathscr{P}(A))$ in the OI case (obtained by iterating the $\mathscr{F}$ in $\mathscr{P}(\mathscr{F}(A))$ and $\mathscr{F}_\Delta(\mathscr{P}(A))$, respectively).

7.2. DEFINITION. Let Σ be an S-sorted alphabet and A a Σ-algebra. Let $n \geqslant 0$ and $s \in S$.

(i) A subset of A_s *is* IO(n) *equational* if it is equational as an element of the $\sqcup$-continuous $D^n(\Sigma)$-algebra $\mathscr{P}(\mathscr{F}^n(A))$.

(ii) A subset of A_s is OI(n) *equational* if it is equational as an element of the Δ-continuous $D^n(\Sigma)$-algebra $\mathscr{F}_\Delta^n(\mathscr{P}(A))$. More generally, for a Δ-continuous Σ-algebra B with $\sqcup$-complete carriers, an element of B_s is OI(n) equational if it is equational as an element of the $D^n(\Sigma)$-algebra $\mathscr{F}_\Delta^n(B)$. ▌

The sort of the elements of $\mathscr{P}(\mathscr{F}^n(A))$ and $\mathscr{F}_\Delta^n(\mathscr{P}(A))$ in which we are interested (i.e., for every $s \in S$, the subsets of A_s) will be denoted by $t_n(s)$. Thus

$$t_0(s) = s \qquad \text{and} \qquad t_{n+1}(s) = \langle \lambda, t_n(s) \rangle.$$

Note that $\mathscr{F}(A)_{\langle \lambda, s \rangle} = A_s$, $\mathscr{P}(A)_s = \mathscr{P}(A_s)$ and $\mathscr{F}_\Delta(B)_{\langle \lambda, s \rangle} = B_s$.

7.3. EXAMPLE. Let Σ be the S-sorted alphabet with $S = \{s\}$, $\Sigma_{\lambda,s} = \{a\}$, and $\Sigma_{ss,s} = \{g\}$ (thus Σ is a ranked alphabet with a constant a and a binary operator g; for the sake of the example we shall use the many-sorted notation). Consider the $S^* \times S$-sorted context-free tree grammar $G = (D(\Sigma), \mathscr{F}, P)$, where $\mathscr{F}_{\lambda,\langle \lambda,s \rangle} = \{Q\}$, $\mathscr{F}_{\langle s,s \rangle,\langle s,s \rangle} = \{F\}$, and the productions are

$$Q \to c_{s,\lambda,s}(F(c_{ss,s,s}(g' \pi_1^{\langle s,s \rangle} \pi_1^{\langle s,s \rangle})) a'),$$

$$F(x_{1,\langle s,s \rangle}) \to F(c_{s,s,s}(x_{1,\langle s,s \rangle} x_{1,\langle s,s \rangle})),$$

and

$$F(x_{1,\langle s,s \rangle}) \to x_{1,\langle s,s \rangle}.$$

Then G^D is a system of regular $D^2(\Sigma)$-equations (cf. Definition 4.5). Thus, for any Σ-algebra A, the solution of Q in the $D^2(\Sigma)$-algebra $\mathscr{P}(\mathscr{F}^2(A))(\mathscr{F}_\Delta^2(\mathscr{P}(A)))$ is an IO(2) equational (OI(2) equational) subset of A. Note that the sort of this solution is $t_2(s) =$

$\langle\lambda, \langle\lambda, s\rangle\rangle$. Without all the confusing sub- and superscripts the grammar G looks like this:

$$Q \to c(F(d)\, a'), \qquad \text{where} \quad d = c(g'\pi_1\pi_1),$$
$$F(x) \to F(c(xx)),$$

and

$$F(x) \to x.$$

Consider the Σ-algebra A with domain a^* (all strings of symbols a), $a_A = a$ and g_A is concatenation. We shall now solve the above equations in the algebras $\mathscr{P}(\mathscr{F}^2(A))$ and $\mathscr{F}_\Delta^2(\mathscr{P}(A))$, using informal but intuitively clear notation. First, in $\mathscr{P}(\mathscr{F}^2(A))$, $F = \{F_n \mid n \geqslant 0\}$ where $F_n \in \mathscr{F}^2(A)$ denotes the function such that, for any $f: A \to A$, $F_n(f) = f^{2^n}$. Since $d: A \to A$ and for every $w \in A$, $d(w) = ww$, $F(d)$ is the set of functions $\{d^{2^n} \mid n \geqslant 0\}$ and $c(F(d)a') = \{d^{2^n}(a) \mid n \geqslant 0\} = \{a^{2^{2^n}} \mid n \geqslant 0\}$. Second, in $\mathscr{F}_\Delta^2(\mathscr{P}(A))$, $F = \bigsqcup_{n\geqslant 0} F_n$, where $F_n \in \mathscr{F}_\Delta^2(\mathscr{P}(A))$ denotes the function such that, for any $f: \mathscr{P}(A) \to \mathscr{P}(A)$, $F_n(f) = f^{2^n}$. Since $d: \mathscr{P}(A) \to \mathscr{P}(A)$ and for every $B \subseteq A$, $d(B) = BB$, $F(d) = \bigsqcup_{n\geqslant 0} F_n(d) = \bigsqcup_{n\geqslant 0} d^{2^n}$ and $c(F(d)a') = \bigsqcup_{n\geqslant 0} d^{2^n}(\{a\}) = \bigcup_{n\geqslant 0} \{a\}^{2^{2^n}} = \{a^{2^{2^n}} \mid n \geqslant 0\}$. Hence the language $\{a^{2^{2^n}} \mid n \geqslant 0\}$ is both an IO(2) equational and OI(2) equational subset of a^*. ∎

Some elementary facts are stated briefly in the following lemma.

7.4. LEMMA.

(1) IO(0) *equational* = OI(0) *equational* = *equational*,

(2) IO(1) *equational* = IO *equational*,
OI(1) *equational* = OI *equational*,

(3) *For tree languages*,
IO(0) *equational* = OI(0) *equational* = *recognizable*,
IO(1) *equational* = IO *tree language*,
OI(1) *equational* = OI *tree language* (*where, for an infinite alphabet, only tree languages over finite subalphabets are considered*).

(4) *For all* $n \geqslant 0$, IO(n) *equational implies* IO($n + 1$) *equational, and*
OI(n) *equational implies* OI($n + 1$) *equational.*

In the next theorem we shall prove an MW-like result for IO(n) equational subsets, which shows that the IO hierarchy is the one intended in [17]. From [21] we know that, for a Σ-algebra A and $s \in S$, a subset of A_s is equational iff it is the Σ-homomorphic image of a recognizable Σ-tree language. From Sections 4 and 5 we know that a subset of A_s is IO equational iff it is the Σ-homomorphic image of the YIELD of a recognizable $D(\Sigma)$-tree language (of sort $\langle\lambda, s\rangle$). The general result will be that a subset of A_s is IO(n) equational iff it is the Σ-homomorphic image of the YIELD^n of a recognizable $D^n(\Sigma)$-tree language (of sort $t_n(s)$). Note that YIELD maps $T_{D(\Sigma),\langle\lambda,s\rangle}$ to $T_{\Sigma,s}$. Hence, by induction, YIELD^n maps $T_{D^n(\Sigma),t_n(s)}$ to $T_{\Sigma,s}$.

7.5. THEOREM. *Let Σ be an S-sorted alphabet, A a Σ-algebra, and h_A the Σ-homomorphism $T_\Sigma \to A$. For any $s \in S$ and $n \geqslant 0$, a subset of A_s is* IO(n) *equational if and only if it is the image under $h_A \circ \mathrm{YIELD}^n$ of a recognizable tree language in $T_{D^n(\Sigma), t_n(s)}$. In particular, a tree language over Σ is* IO(n) *equational iff it is* $\mathrm{YIELD}^n(L)$ *for some recognizable tree language L over $D^n(\Sigma)$ (of appropriate sort).* ∎

Proof. If h_n denotes the unique $D^n(\Sigma)$-homomorphism $T_{D^n(\Sigma)} \to \mathscr{F}^n(A)$, then the solution of any system of regular $D^n(\Sigma)$-equations in $\mathscr{P}(\mathscr{F}^n(A))$ is the h_n image of its solution in $\mathscr{P}(T_{D^n(\Sigma)})$, which is a recognizable $D^n(\Sigma)$-tree language (see Lemma 5.2 and Section 2.3). Thus it suffices to show that $h_n = h_A \circ \mathrm{YIELD}^n$ on any sort $t_n(s)$. We show this by induction on n. For $n = 0$ there is nothing to prove. For $n = 1$ it is clear that $h_1 = h_{\mathscr{F}(A)} = \mathrm{derop}_A \circ \mathrm{YIELD}$. Moreover $\mathrm{derop}_A(t) = h_A(t)$ for each t of sort $\langle \lambda, s \rangle$. Hence $h_1 = h_A \circ \mathrm{YIELD}$. Suppose that the statement is true for n. We now apply the case $n = 1$ to the case $n + 1$ by taking $D^n(\Sigma)$ instead of Σ and $\mathscr{F}^n(A)$ instead of A, as follows (note that $h_n = h_{\mathscr{F}^n(A)}$):

$$\begin{aligned} h_{n+1} &= \mathrm{derop}_{\mathscr{F}^n(A)} \circ \mathrm{YIELD} \\ &= h_n \circ \mathrm{YIELD} \qquad \text{on } t_{n+1}(s) \\ &= h_A \circ \mathrm{YIELD}^n \circ \mathrm{YIELD} = h_A \circ \mathrm{YIELD}^{n+1}. \end{aligned}$$

This proves the statement and the theorem. ∎

We note that originally the proof of this theorem was much longer. The present short proof is due to Damm [44].

As an immediate consequence of Theorem 7.5 we state the following MW-like corollary.

7.6. COROLLARY. *Let Σ be an S-sorted alphabet, A a Σ-algebra, $n \geqslant 0$, and $s \in S$. A subset of A_s is* IO(n) *equational iff it is the Σ-homomorphic image of an* IO(n) *equational tree language over Σ (of sort s).* ∎

7.7. EXAMPLE. Consider Example 7.3. It is left to the reader to show that the IO(2) equational subset $\{a^{2^{2^n}} \mid n \geqslant 0\}$ is $h_A(\mathrm{YIELD}(L))$, where L is the IO $D(\Sigma)$-tree language generated by the nonterminal Q of the grammar G. For instance, without sub- and superscripts, $Q \overset{*}{\Rightarrow} c(c(dd)a')$ and $h_A(\mathrm{YIELD}(c(c(dd)a'))) = h_A(g(g(aa)g(aa))) = a^4$. ∎

As an example we now consider the IO hierarchy of string languages.

Let Ω be an ordinary (finite) alphabet. There are two well-known ways of considering Ω^* as a Σ-algebra. First, let Ω^c be the ranked alphabet determined by ${\Omega_0}^c = \Omega \cup \{e\}$ and ${\Omega_2}^c = \{c\}$, where e and c are new symbols. Ω^* is viewed as an Ω^c-algebra, denoted by ${\Omega_c}^*$, by defining e to be the empty string λ, every symbol in Ω to be itself, and c to be string concatenation. The unique Ω^c-homomorphism $T_{\Omega^c} \to {\Omega_c}^*$ is called *yield* (cf. Example 2.2.3; note that "yield" differs from the usual one by the fact that $\mathrm{yield}(e) = \lambda$). Second, let Ω^m be the (monadic) ranked alphabet determined by ${\Omega_0}^m = \{e\}$ and ${\Omega_1}^m = \Omega$. Ω^* is viewed as an Ω^m-algebra, denoted by ${\Omega_m}^*$, by defining $e = \lambda$ and, for $w \in \Omega^*$ and $a \in \Omega$, $a(w) = aw$ (thus $a \in {\Omega_1}^m$ is interpreted as left concatenation with a). Since the

unique Ω^m-homomorphism $T_{\Omega^m} \to \Omega_m^*$ is clearly an isomorphism, we shall not distinguish between T_{Ω^m} and Ω_m^* as Ω^m-algebras.

Thus an IO hierarchy of string languages over Ω can be built up in two ways: by viewing Ω^* either as an Ω^c-algebra or as Ω^m-algebra. It is well known that the equational subsets of Ω_c^* are the context-free languages over Ω ([21]; e causes no problem).

Let us now define IO (and OI) string languages. It should be clear to the reader that our definition is equivalent to the usual one in [12].

7.8. DEFINITION. Let Ω be an alphabet. A language over Ω is called an IO *string language* (OI *string language*) over Ω if it is the yield of an IO tree language (OI tree language) over Ω^c. ∎

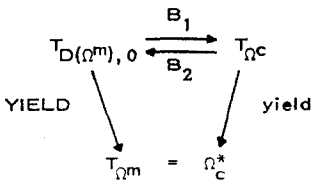
It follows directly from this definition and Corollaries 5.11 and 5.18 that the IO (OI) equational subsets of Ω_c^* are the IO (OI) string languages over Ω. Thus the hierarchy of IO(n) equational languages in Ω_c^* starts with the context-free and the IO string languages. We now consider Ω_m^*.

7.9. THEOREM. *Let Ω be an alphabet. For $n = 0$, 1, and 2 the* IO(n) *equational subsets of Ω_m^** are the regular, *context-free, and* IO *string languages, respectively.*

Proof. For $n = 0$ the statement should be clear from the isomorphism between Ω_m^* and T_{Ω^m}. For $n = 1, 2$ we know from Theorem 7.5 that an IO(n) equational subset of T_{Ω^m} is the YIELD of an IO($n - 1$) equational subset of $T_{D(\Omega^m),0}$. Hence the IO(1) equational subsets of Ω_m^* are the YIELDs of recognizable tree languages in $T_{D(\Omega^m),0}$, and the IO(2) equational subset of Ω_m^* are the YIELDs of IO tree languages in $T_{D(\Omega^m),0}$.

Consider, informally, some tree in $T_{D(\Omega^m),0}$. Then one notices that it contains in general many superfluous subtrees (w.r.t. YIELD), due to the fact that, since all elements of Ω^m have rank 1 or 0, trees in $T_{\Omega^m}(X)$ can contain at most one variable. In fact, one may see that any tree t_1 in T_{Ω^m} is YIELD of a tree t_2 in $T_{D(\Omega^m),0}$ in which only the symbols from $\Omega \cup \{e, c_{1,1}, c_{1,0}\}$ are used. Moreover, for such a tree t_2, when $c_{1,1}$ and $c_{1,0}$ are identified with c, YIELD(t_2) = yield(t_2) = t_1.

Formally we shall show the existence of two mappings B_1: $T_{D(\Omega^m),0} \to T_{\Omega^c}$ and B_2: $T_{\Omega^c} \to T_{D(\Omega^m),0}$ such that yield($B_1(t)$) = YIELD(t) and YIELD($B_2(t)$) = yield(t). Thus



Moreover we shall show that B_1 and B_2 preserve the properties of being a recognizable tree language and being an IO tree language. From this the theorem can be proved as follows. For $n = 1, 2$, if L is an IO(n) equational subset of Ω_m^*, then L = YIELD(R) for some IO($n - 1$) equational tree language R in $T_{D(\Omega^m),0}$. Hence L = yield($B_1(R)$) and

$B_1(R)$ is an IO($n-1$) equational tree language in T_{Ω^c}. Therefore, L is an IO($n-1$) equational subset of $\Omega_c{}^*$, i.e., for $n=1$ a context-free language and for $n=2$ an IO string language. Conversely, for $n=1, 2$, if L is an IO($n-1$) equational subset of $\Omega_c{}^*$, the $L=$ yield(R) for some IO($n-1$) equational tree language R over Ω^c. Hence $L=$ YIELD($B_2(R)$) and $B_2(R)$ is an IO($n-1$) equational tree language over $D(\Omega^m)$. Therefore L is an IO(n) equational subset of $\Omega_m{}^*$.

We now show the existence of B_1 and B_2. B_1 will be realized as a linear deterministic bottom-up tree transducer mapping from $D(\Omega^m)$ to Ω^c (in fact $\overline{D(\Omega^m)}$ to $\overline{\Omega^c}$). It then follows from Section 6 and Theorem 7.1 that B_1 preserves the recognizable and the IO tree languages. More precisely, B_1 only exists for each finite subalphabet of $D(\Omega^m)$ and this clearly suffices for our purpose. We shall, however (as we did in Section 6), construct an infinite B_1 for $D(\Omega^m)$ and leave it to the reader to restrict B_1 to a finite transducer for each finite subalphabet of $D(\Omega^m)$. For notation, see Section 6.

B_1 has states $Q=\mathbb{N}$ and final states $F=Q$. The family of mappings $\langle B_n\rangle_{n\in\mathbb{N}}$ is specified as follows:

(i) for $1\leqslant i\leqslant n$, $B_0(\pi_i{}^n)=(i, e)$;

(ii) $B_0(e')=(0, e)$;

(iii) for $f\in\Omega$, $B_0(f')=(1, f)$;

(iv) for $n, k\geqslant 0$, $B_{n+1}(c_{n,k})$ is a mapping $Q^{n+1}\to Q\times T_{\Omega^c}(X_{n+1})$ such that, for $j, i_1, \ldots, i_n\in N$,

$$\begin{aligned} B_{n+1}(c_{n,k})(j, i_1, \ldots, i_n) &= (j, x_1) && \text{if } j=0\\ &= (i_j, c(x_1x_{j+1})) && \text{if } 1\leqslant j\leqslant n\\ &= \text{arbitrary otherwise.} \end{aligned}$$

We note that the elements of $T_{\Omega^m}(X)$ can be identified with the strings in $\Omega^*\cup\Omega^*X$. We leave it to the reader to check that (with this identification) for $t\in T_{D(\Omega^m)}$, $i\in\mathbb{N}$, and $t'\in T_{\Omega^c}$, if $\bar{B}_0(t)=(i, t')$ then either $i=0$ and YIELD(t) = yield(t'), or $i\geqslant 1$ and YIELD(t) = yield(t') $\cdot x_i$. Hence, for all $t\in T_{D(\Omega^m),0}$, yield($B_1(t)$) = YIELD(t).

The mapping B_2: $T_{\Omega^c}\to T_{D(\Omega^m),0}$ is easy to describe: for $s\in T_{\Omega^c}$, $B_2(s)=c_{1,0}(te')$, where t is obtained from s by relabeling c as $c_{1,1}$, each $f\in\Omega$ as f', and e as $\pi_1{}^1$. It is easy to see that $t\in T_{D(\Omega^m),1}$ and YIELD(t) = yield(s) $\cdot x_1$. Hence $B_2(s)\in T_{D(\Omega^m),0}$ and YIELD($B_2(s)$) = yield(s). It should be obvious that B_2 preserves the recognizable and IO tree languages (cf. Theorem 7.1). This concludes the proof of the theorem. ∎

Note that, if, as we conjecture, the IO(n) equational tree languages are closed under deterministic bottom-up tree transducer mappings, then the proof of the previous theorem shows that, for all n, the IO(n) equational subsets of $\Omega_m{}^*$ equal the IO($n-1$) equational subsets of $\Omega_c{}^*$.

We conclude this section by considering the OI case. This case can be treated completely analogous to the IO case, using infinite trees (with $+$) rather than tree languages. Since infinite trees are more or less outside the scope of this paper, no detailed proofs will be given. For notation and some properties of infinite trees we refer to the last part of

Section 5. The generalization to the many-sorted case is understood (cf. [15]). Recall that, for any S-sorted alphabet Σ, $\Sigma^+ = \Sigma \cup \{+_s \mid s \in S\}$, SET denotes the homomorphism $CT_{\Sigma^+} \to \mathscr{P}(T_\Sigma)$, and YIELD denotes the homomorphism $CT_{D(\Sigma)^+} \to CT_{\Sigma^+}(X_S)$. Note that YIELD maps $CT_{D(\Sigma)^+,\langle\lambda,s\rangle}$ into $CT_{\Sigma^+,s}$. Hence YIELD^n maps $CT_{D^n(\Sigma)^+,t_n(s)}$ into $CT_{\Sigma^+,s}$. Recall also that the solution of a system of regular (context-free) Σ^+-equations in CT_{Σ^+} is called a recognizable (context-free) infinite tree.

The following MW-like result for OI(n) equational elements is the analog of Theorem 7.5.

7.10. THEOREM. *Let Σ be an S-sorted alphabet, B a Δ-continuous Σ-algebra with $\sqcup$-complete carriers, and h_B the unique Δ-continuous $\perp$-preserving Σ^+-homomorphism $CT_{\Sigma^+} \to B$. For any $s \in S$ and $n \geqslant 0$, an element of B_s is* OI(n) *equational if and only if it is the image under $h_B \circ \text{YIELD}^n$ of a recognizable infinite tree in $CT_{D^n(\Sigma)^+,t_n(s)}$.*

Proof. The proof is completely analogous to the proof of Theorem 7.5, using $\mathscr{F}_\Delta$ rather than $\mathscr{F}$, Δ-continuous $D^n(\Sigma)^+$-algebras rather than $D^n(\Sigma)$-algebras, Δ-continuous $\perp$-preserving $D^n(\Sigma)^+$-homomorphisms rather than $D^n(\Sigma)$-homomorphisms, $CT_{D^n(\Sigma)^+}$ rather than $T_{D^n(\Sigma)}$, and extending all $D^n(\Sigma)$-algebras with $\sqcup$-complete carriers to $D^n(\Sigma)^+$-algebras by defining $+$ to be $\sqcup$. ∎

As a particular case of this theorem we obtain the following result for OI(n) equational subsets of a Σ-algebra.

7.11. THEOREM. *Let Σ be an S-sorted alphabet, A a Σ-algebra, and h_A the Σ-homomorphism $T_\Sigma \to A$. For $s \in S$ and $n \geqslant 0$, a subset of A_s is* OI(n) *equational if and only if it is the image under $h_A \circ \text{SET} \circ \text{YIELD}^n$ of a recognizable infinite tree in $CT_{D^n(\Sigma)^+,t_n(s)}$. In particular, a tree language over Σ is* OI(n) *equational iff it is* $\text{SET}(\text{YIELD}^n(t))$ *for some recognizable infinite tree over $D^n(\Sigma)^+$ (of appropriate sort).*

Proof. Immediate from Theorem 7.10 by the fact that, for $B = \mathscr{P}(A)$, $h_B = h_A \circ \text{SET}$. ∎

An immediate consequence of this theorem is the following MW-like corollary.

7.12. COROLLARY. *Let Σ be an S-sorted alphabet, A a Σ-algebra, $n \geqslant 0$ and $s \in S$. A subset of A_s is* OI(n) *equational iff it is the Σ-homomorphic image of an* OI(n) *equational tree language over Σ (of sort s).* ∎

7.13. EXAMPLE. Consider Example 7.3. It is left to the reader to show that the OI(2) equational subset $\{a^{2^{2^n}} \mid n \geqslant 0\}$ is $h_A(\text{SET}(\text{YIELD}(t)))$, where t is the infinite context-free $D(\Sigma)^+$-tree determined by the nonterminal Q of G. For instance

$$\begin{aligned} &h_A(\text{SET}(\text{YIELD}(c(+(+(\cdots c(dd))d)\, a')))) \\ &\quad = h_A(\text{SET}(+(+(\cdots g(g(aa)\, g(aa)))\, g(aa)))) \\ &\quad = h_A(\{g(aa), g(g(aa)\, g(aa)), \ldots\}) = \{a^2, a^4, \ldots\}. \end{aligned}$$ ∎

To contrast again the IO and OI cases we state the next IO result, to be compared with Theorem 7.11.

7.14. THEOREM. *Let Σ be an S-sorted alphabet, A a Σ-algebra, and $h_A\colon T_\Sigma \to A$. For any $n \geqslant 0$ and $s \in S$, a subset of A_s is* IO(n) *equational iff it is the image under $h_A \circ \mathrm{YIELD}^n \circ \mathrm{SET}$ of a recognizable infinite tree over $D^n(\Sigma)^+$ (of sort $t_n(s)$).*

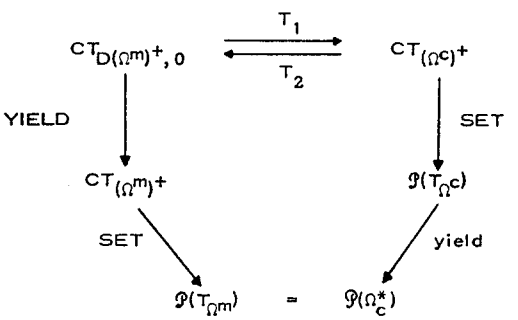
Proof. Immediate from Theorem 7.5 and the fact that SET is a Δ-continuous $\perp$-preserving $D^n(\Sigma)^+$-homomorphism from $CT_{D^n(\Sigma)^+}$ into $\mathcal{P}(T_{D^n(\Sigma)})$. ∎

Thus IO(n) equational tree languages are obtained from the recognizable infinite trees in $CT_{D^n(\Sigma)^+}$ by application of $\mathrm{YIELD}^n \circ \mathrm{SET}$, and the OI($n$) equational tree languages by application of $\mathrm{SET} \circ \mathrm{YIELD}^n$ (this generalizes the diagram at the end of Section 5).

Next we consider the OI hierarchies of string languages. As in the IO case, consider a string alphabet Ω and the two possible algebras $\Omega_c{}^*$ and $\Omega_m{}^*$. Clearly, the hierarchy of OI(n) equational languages in $\Omega_c{}^*$ starts with the context-free and the OI string languages (see Definition 7.8). We now show that the hierarchy of OI(n) equational subsets of $\Omega_m{}^*$ starts with the regular, the context-free, and the OI string languages (cf. [42]).

7.15. THEOREM. *Let Ω be an alphabet. For $n = 0, 1, 2$, the* OI(n) *equational subsets of $\Omega_m{}^*$ are the regular, context-free, and* OI *string languages, respectively.*

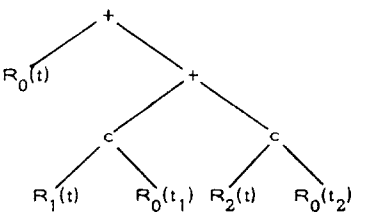
Proof. For $n = 0$ the statement is clear. For $n = 1, 2$ we shall only sketch a possible proof. From Theorem 7.10 it can be seen that the OI(1) equational subsets of $\Omega_m{}^*$ are the $\mathrm{SET} \circ \mathrm{YIELD}$s of recognizable infinite trees over $D(\Omega^m)^+$ (of sort 0), whereas the OI(2) equational subsets of $\Omega_m{}^*$ are the $\mathrm{SET} \circ \mathrm{YIELD}$s of context-free infinite trees over $D(\Omega^m)^+$ (of sort 0). Analogously to the proof of Theorem 7.9 it would suffice to have mappings T_1 and T_2 such that the following diagram commutes



and such that T_1 and T_2 preserve recognizability and context freeness of infinite trees. As in the IO case, the existence of T_2 is obvious. For T_1, think of a deterministic top-down tree transducer (see [29]) working on an infinite tree. Taking the join of all initial pieces

of output, one obtains an infinite tree as output of the transducer. Using arguments similar to the language case, one can prove that, in general, the classes of recognizable and context-free infinite trees are closed under deterministic top-down tree transducers (note that copying and deletion are no problem since the infinite tree is "generated by a deterministic grammar"). It should be clear that a deterministic topdown tranducer R, realizing T_1, can be constructed (for any finite subalphabet of $D(\Omega^m)$). R has the set of states $\mathbb{N}$ and, denoting the output of R started in state i on input t by $R_i(t)$, it has the property that, for any tree t in $CT_{D(\Omega^m)+}$, $\mathrm{SET}(\mathrm{YIELD}(t)) = \mathrm{yield}(\mathrm{SET}(R_0(t))) \cup \bigcup_{i \geqslant 1} (\mathrm{yield}(\mathrm{SET}(R_i(t))) \cdot x_i)$. For instance, the rule of R for the symbol $c_{2,k}$ and state 0 might look like

$R_0(c_{2,k}(tt_1t_2)) =$



The detailed construction of R is left to the reader. ∎

Finally we consider the OI hierarchy for an arbitrary nondeterministic monadic algebra, i.e., a domain with a finite number of nondeterministic unary operations. Let Ω be an alphabet, let D be a set and let, for each $f \in \Omega$, f_D be a mapping $D \to \mathcal{P}(D)$, or equivalently a subset of $D \times D$. Consider first the $\sqcup$-continuous Ω^c-algebra A with domain $\mathcal{P}(D \times D)$, such that for $f \in \Omega$, $f_A = f_D$, e_A is the identity mapping, i.e., $e_A = \{(d, d) \mid d \in D\}$, and c_A is composition of binary relations. Since there is a (unique $\sqcup$-continuous Ω^c-homomorphism $\mathcal{P}(\Omega_c{}^*) \to \mathcal{P}(D \times D)$ (see [11]), the equational (OI equational) elements of $\mathcal{P}(D \times D)$ are the Ω^c-homomorphic images of the context-free languages (OI string languages) over Ω. It was shown in [11, Sections 4 and 5] that these are the relations computed by the nondeterministic recursive monadic program schemes in D (the relations computed by the so-called nondeterministic procedure parameter schemes in D, respectively). Consider now the nondeterministic Ω^m-algebra D with domain D, nondeterministic monadic operations f_D and some "input element" $e_D \in D$. It follows from Theorem 7.15 and Corollary 7.12 (adapted in the obvious way to the case of a nondeterministic algebra) that, for $n = 0$, 1, and 2, the OI(n) equational subsets of D are the Ω^m-homomorphic images of the regular, context-free, and OI string languages in $\Omega_m{}^*$, respectively. From a comparison of the Ω^m-homomorphism $\mathcal{P}(\Omega_m{}^*) \to \mathcal{P}(D)$ with the Ω^c-homomorphism $\mathcal{P}(\Omega_c{}^*) \to \mathcal{P}(D \times D)$ it easily follows that these equational subsets are the images of e_D under the Ω^c-homomorphic images of the regular, context-free, and OI string languages, i.e., they are the sets computed from the input e_D by the nondeterministic Ianov schemes, the nondeterministic recursion schemes, and the nondeterministic procedure parameter schemes, respectively (see [11, Sections 3, 4, and 5]). This shows that these three classes of program schemes correspond in a natural algebraic way to the hierarchy of regular, context-free, and OI string languages.

CONCLUSION

We have shown that the fixed point approach to formal language theory and the theory of programs can be used to explain the differences between IO (call by value) and OI (call by name) computation. Moreover, in the framework of continuous algebras, we gained a better insight into the various more or less well-known Mezei-and-Wright like results in this area.

The fixed point characterization of context-free tree languages implied that IO tree languages are YIELDs of recognizable second level tree languages, and that OI tree languages are SET $\circ$ YIELDs of recognizable second level infinite trees (with $+$). It might be interesting to prove more results about IO and OI tree languages using either the fixed point characterization itself (see, for instance, Section 4 and [8]) or its implications (see, for instance, Section 6). To treat the OI case it might be advantageous to develop a theory of infinite trees (see [7, 10, 15]).

Not much is known about the hierarchies defined in Section 7. We conjecture that all the IO tree language classes in the hierarchy are closed under deterministic bottom-up tree transducers and that all the OI tree language classes are closed under linear (top-down or bottom-up) tree transducers. For OI, together with the obvious closure under OI substitution, a result in [2, Theorem 3.2.12] would then imply that all the OI string language classes are (substitution closed) full AFLs. Other questions concerning the hierarchies are: what is the relationship to the tree transducer hierarchy [2, 24], what "nondeterministic power" is present in the IO hierarchy, and what "copying power" in the OI hierarchy?

As regards programscheme theory, the remarks in Section 5, together with the incomparability of the classes of IO and OI tree languages, show that our classes of nondeterministic IO (call by value) and OI (call by name) recursive program schemes (without tests) are incomparable with respect to program scheme equivalence. About deterministic recursive program schemes with tests not very much is known. It is easy to see that every deterministic call by value scheme is equivalent to a deterministic call by name scheme, which is forced to evaluate its arguments by some trivial test (provided these are present). It can also be proved (see [26]) that every deterministic call by name scheme is equivalent to a nondeterministic call by value scheme (which guesses which of its arguments it actually will need). The precise relationship between the classes of deterministic call by value and call by name schemes remains open. It is also an open question whether a useful "universal interpretation" exists for deterministic schemes with tests.

Hopefully this paper will be of some help in the solution of these problems.

REFERENCES

1. E. ASHCROFT, Z. MANNA, AND A. PNUELI, Decidable properties of monadic functional schemas, *J. Assoc. Comput. Mach.* **20** (1973), 489–499.
2. B. S. BAKER, "Tree Transductions and Families of Tree Languages," Ph.D. Thesis, Harvard Univ., Report TR-9-73, 1973.

3. H. BEKIĆ, "Definable Operations in General Algebras, and the Theory of Automata and Flowcharts," IBM Laboratory, Vienna, 1969.
4. G. BIRKHOFF AND J. D. LIPSON, Heterogeneous algebras, *J. Combinatorial Theory* 8 (1970), 115–133.
5. A. BLIKLE, Equational languages, *Inform. Contr.* 12 (1972), 134–147.
6. P. M. COHN, "Universal Algebra," Harper and Row, New York, 1965.
7. B. COURCELLE, Recursive schemes, algebraic trees and deterministic languages, *in* "15th Annual Symposium on Switching & Automata Theory, 1974," pp. 52–62.
8. P. J. DOWNEY, "Formal Languages and Recursion Schemes," Harvard Univ., Report TR-16-74, 1974.
9. J. ENGELFRIET, Bottom-up and top-down tree transformations—a comparison, *Math. Systems Theory* 9 (1975), 198–231.
10. J. ENGELFRIET, A note on infinite trees, *Inform. Proc. Lett.* 1 (1972), 229–232.
11. J. ENGELFRIET, "Simple Program Schemes and Formal Languages," Lecture Notes in Computer Science 20, Springer–Verlag, Berlin, 1974.
12. M. J. FISCHER, "Grammars with Macro-like Productions," Doctoral Dissertation, Harvard Univ., 1968 (see also 9th SWAT, pp. 131–142).
13. S. J. GARLAND AND D. C. LUCKHAM, Program schemes, recursion schemes and formal languages, *J. Comput. System Sci.* 7 (1973), 119–160.
14. S. GINSBURG AND H. G. RICE, Two families of languages related to ALGOL, *J. Assoc. Comput. Mach.* 9 (1962), 350–371.
15. J. A. GOGUEN AND J. W. THATCHER, Initial algebra semantics, *in* "15th Annual Symposium on Switching and Automata Theory, 1974," pp. 63–77.
16. J. A. GOGUEN, J. W. THATCHER, E. G. WAGNER, AND J. B. WRIGHT, Initial algebra semantics, *J. Assoc. Comput. Mach.* 24 (1977), 68–95.
17. T. S. E. MAIBAUM, A generalized approach to formal languages, *J. Comput. System Sci.* 8 (1974), 409–439.
18. Z. MANNA, "Mathematical Theory of Computation," McGraw–Hill, New York, 1974.
19. Z. MANNA, S, NESS, AND J. VUILLEMIN, Inductive methods for proving properties of programs, *Comm. ACM* 16 (1973), 491–502.
20. J. MCCARTHY, A basis for a mathematical theory of computation, *in* "Computer Programming and Formal Systems" (P. Braffort and D. Hirschberg, Eds.), pp. 33–70, North–Holland, Amsterdam, 1963.
21. J. MEZEI AND J. B. WRIGHT, Algebraic automata and context-free sets, *Inform. Contr.* 11 (1967), 3–29.
22. M. NIVAT, Langages algébriques sur le magma libre et sémantique des schémas de programme, *in* "Automata, Languages and Programming" (M. Nivat, Ed.), pp. 293–307, North–Holland, Amsterdam, 1973.
23. M. NIVAT, On the interpretation of recursive program schemes, *Symposia Mat.* 15 (1975), 255–281.
24. W. F. OGDEN AND W. C. ROUNDS, Composition of n tree transducers, *in* "4th Annual ACM Symposium on Theory of Computing, Denver, Colorado, 1972," pp. 198–206.
25. W. P. DE ROEVER, Recursion and parameter mechanisms: an axiomatic approach, *in* "Automata, Languages and Programming" (J. Loeckx, Ed.), Lecture Notes in Computer Science 14, Springer–Verlag, Berlin, 1974.
26. W. P. DE ROEVER, First order reduction of call-by-name to call-by-value, *in* "International Symposium on Proving and Improving Programs, Arc-et-Senans, 1975."
27. G. F. ROSE, An extension of ALGOL-like languages, *Comm. ACM* 7 (1964), 52–71.
28. B. K. ROSEN, Tree-manipulating systems and Church–Rosser theorems, *J. Assoc. Comput. Mach.* 20 (1973), 160–188.
29. W. C. ROUNDS, Mappings and grammars on trees, *Math. Systems Theory* 4 (1970), 257–287.

30. W. C. ROUNDS, Tree-oriented proofs of some theorems on context-free and indexed languages, *in* "Second Annual Symposium on Theory of Computing, Northampton, Mass., 1970," pp. 109–116.
31. D. SCOTT, Data types as lattices, *SIAM J. Comp.* **5** (1976), 522–587.
32. D. SCOTT, "Outline of a Mathematical Theory of Computation," Technical Monograph PRG-2, Oxford Univ. 1970.
33. J. W. THATCHER, Generalized^2 sequential machine maps, *J. Comput. System Sci.* **4** (1970), 339–367.
34. J. W. THATCHER, "Generalized^2 Sequential Machine Maps," IBM Report RC 2466, 1969.
35. J. W. THATCHER, Tree automata: an informal survey, *in* "Currents in the Theory of Computing" (A.V. Aho, Ed.), pp. 143–172, Prentice–Hall, 1973.
36. J. W. THATCHER AND J. B. WRIGHT, Generalized finite automata theory with an application to a decision problem of second-order logic, *Math. Systems Theory* **2** (1968), 57–81.
37. R. TURNER, An infinite hierarchy of term languages—an approach to mathematical complexity, *in* "Automata, Languages and Programming" (M. Nivat, Ed.), pp. 593–608, North–Holland, Amsterdam, 1973.
38. J. VUILLEMIN, "Syntaxe, sémantique et axiomatique d'un langage de programmation," These, Université Paris VI, 1974.
39. E. G. WAGNER, Languages for defining sets in arbitrary algebras, *in* "12th Annual Symposium on Switching and Automata Theory, 1971," pp. 192–201.
40. M. WAND, A concrete approach to abstract recursive definitions, *in* "Automata, Languages and Programming" (M. Nivat, Ed.) pp. 331–344, North–Holland, Amsterdam, 1972.
41. M. WAND, "Mathematical Foundations of Formal Language Theory," Massachusetts Institute of Technology, Report MAC TR-108, 1973.
42. M. WAND, An algebraic formulation of the Chomsky hierarchy, *in* "Category Theory Applied to Computation and Control," pp. 209–213, Lecture Notes in Computer Science 25, Springer–Verlag, Berlin, 1975.
43. G. BOUDOL, Thèse de 3ème cycle, Paris VII.
44. W. DAMM, Higher type program schemes and their tree languages, *in* "4th Colloquium on Automata, Languages and Programming, Turku, 1977."